# ENVIRONMENTAL DISTURBANCE MODELING FOR LARGE INFLATABLE SPACE STRUCTURES

THESIS

Donald J. Davis, Captain, USAF

AFIT/GSO/ENY/01M-02

**DEPARTMENT OF THE AIR FORCE**
**AIR UNIVERSITY**

# AIR FORCE INSTITUTE OF TECHNOLOGY

**Wright-Patterson Air Force Base, Ohio**

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

# ENVIRONMENTAL DISTURBANCE MODELING FOR LARGE INFLATABLE SPACE STRUCTURES

THESIS

Presented to the Faculty

Department of Aeronautics and Astronautics

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Space Operations

Donald J. Davis, B.S.

Captain, USAF

March 2001

# ENVIRONMENTAL DISTURBANCE MODELING FOR

# LARGE INFLATABLE SPACE STRUCTURES

Donald J. Davis, B.S.
Captain, USAF

Approved:

_____     _____
Gregory S. Agnes (Chair)                    Date   6 Mar 01

_____     _____
William E. Wiesel Jr. (Member)              Date   6 Mar 01

_____     _____
Steven G. Tragesser (Member)               Date   05 MAR 01

*Acknowledgements*

I would like to express my gratitude to my faculty advisor, Maj. Greg Agnes, for his guidance and input throughout the entire thesis effort. Thank you for providing direction when I began to stray off course. Also, I must thank my reading committee, Dr. William Wiesel and Dr. Steve Tragesser, for explaining the finer points of orbital mechanics to me and providing feedback on my research effort.

I am also indebted to my classmates, Maj. Dayne Cook and Capt Jack Oldenberg who also provided valuable support and feedback during the course of developing this product. We spent many hours bouncing ideas off of each other and learning the intricacies of LaTeX 2e.

Finally, but most importantly, I need to thank my wife          for her patience during the many weekends that I spent more time with my computer than with her. Thank you for being so understanding.

<div align="right">

Donald J. Davis

</div>

# Table of Contents

# List of Figures

# List of Tables

# List of Symbols

*English Symbols*

| Symbol | Definition (units) |
|---|---|
| A | Area $(m^2)$ |
| $A_p$ | Planetary Geomagnetic Index |
| a | Semi-major Axis $(km)$ |
| $\vec{a_g}$ | Gravitational Acceleration $(m/s^2)$ |
| $\vec{B_o}$ | Earth's Homogeneous Magnetic Field |
| $C^{ab}$ | Transformation Matrix from $[\hat{a}]$ to $[\hat{b}]$ |
| $C_d$ | Coefficient of Drag |
| c | Speed of Light $(m/s)$ |
| $cm$ | Center of Mass |
| $cp$ | Center of Pressure |
| $c_p$ | Specific Heat $(J/kg/K)$ |
| $cte$ | Coefficient of Thermal Expansion $(K^{-1})$ |
| E | Eccentric Anomaly $(rad)$ |
| e | Eccentricity |
| $f_{10.7}$ | 10.7cm Solar Flux $(10^{-22}W/m^2/cycle/sec)$ |
| f | Force $(N)$ |
| h | Orbital Altitude $(km)$ |
| H | Solar Constant $(W/m^2)$ |
| $H_e$ | Earth Constant $(W/m^2)$ |
| $H_o$ | Scale Height $(km)$ |
| $H_s$ | Solar Flux $(W/m^2)$ |
| i | Inclination |

| | |
|---|---|
| $i_c$ | Current ($amps$) |
| $I^c$ | Inertia about Center of Mass ($kgm^2$) |
| k | Thermal Conductivity ($W/m/K$) |
| $m$ | Mass ($kg$) |
| $\hat{n}$ | Surface Normal Vector |
| P | Pressure ($Pa$) |
| $R_e$ | Earth Radius ($km$) |
| $R_s$ | Sun Radius ($km$) |
| $\vec{r_{es}}$ | Earth-Sun Position Vector |
| $\vec{r_{sat}}$ | Satellite ECI Position Vector |
| $\vec{r_{sve}}$ | Satellite-Earth Position Vector |
| $\vec{r_{svs}}$ | Satellite-Sun Position Vector |
| $S_i$ | Solar Intensity Coefficient |
| $T$ | Temperature (K) |
| $T_{exo}$ | Exospheric Temperature (K) |
| $T_o$ | Time of Perigee Passage |
| $t$ | Time ($days$) |
| $U_g$ | Gravitational Potential ($m^2/s^2$) |
| $\vec{v}$ | Velocity ($m/s$) |

*Greek Symbols*

| Symbol | Definition (units) |
|---|---|
| $\alpha$ | Coefficient of Absorbtion |
| $\beta$ | Coefficient of Reflection |
| $\Delta T$ | Temperature Gradient (K) |
| $\epsilon$ | Thermal Emissivity |

| | |
|---|---|
| $\varepsilon$ | Obliquity of the Ecliptic |
| $\eta$ | Sun-Earth-Satellite Angle (*degrees*) |
| $\theta$ | Angle of Incidence (*degrees*) |
| $\mu$ | Earth Gravitational Parameter ($km^3/s^2$) |
| $\mu_e$ | Earth Magnetic Moment ($Gauss - cm^3$) |
| $\rho$ | Mass Density ($kg/m^3$) |
| $\rho_e$ | Apparent Angular Radius of Earth (*degrees*) |
| $\rho_s$ | Apparent Angular Radius of Sun (*degrees*) |
| $\Psi$ | Earth-Satellite-Sun Angle (*degrees*) |
| $\sigma$ | Stefan-Boltzman Constant ($W/m^2/K^4$) |
| $\upsilon$ | True Anomaly (*degrees*) |
| $\varphi$ | Coefficient of Specular Reflection |
| $\Omega$ | Right Ascension of the Ascending Node (*degrees*) |
| $\omega$ | Argument of Perigee (*degrees*) |

# List of Abbreviations

| Abbreviation | Definition |
|---|---|
| ARISE | Advanced Radio Interferometry between Space and Earth |
| AU | Astronomical Unit |
| ECI | Earth Centered Inertial |
| EUV | Extreme Ultra-Violet |
| IAE | Inflatable Antenna Experiment |
| IR | Infrared |
| JD | Julian Date |
| JPL | Jet Propulsion Laboratory |
| LEO | Low Earth Orbit |
| NASA | National Aeronautics and Space Administration |
| NGST | Next Generation Space Telescope |
| NOAA | National Oceanic and Atmospheric Administration |
| SRP | Solar Radiation Pressure |
| RF | Radio Frequency |
| TES | Thermo-Elastic Shock |
| UARS | Upper Atmosphere Research Satellite |
| UT | Universal Time |
| UV | Ultra-Violet |

AFIT/GSO/ENY/01M-02

## *Abstract*

Tightening space budgets and stagnating spacelift capabilities are driving the Air Force and other space agencies to focus on inflatable technology as a reliable, inexpensive means of deploying large structures in orbit. Recent improvements in rigidization techniques make the use of these inflatable structures feasible for a growing number of missions. For many of these missions, the primary design requirement is dimensional accuracy of the structure. Finite element analysis offers a means of predicting structural behavior in orbit. The analysis requires knowledge of external loads. This thesis examines the environmental disturbances which act upon large, orbiting structures. Calculations are made on a base model to relate the torques generated by these disturbances to the orbital altitude. This facilitates identification of the critical loads for large, inflatable structures. An environmental disturbance model is then developed in MATLAB. The model calculates the critical loads on each element of a faceted structure as it propagates through its orbit. A basic structure is defined and entered into the model. Results and analysis for various orbits are presented to verify accuracy of the code and validate the derived torque-altitude relationships.

# ENVIRONMENTAL DISTURBANCE MODELING FOR LARGE INFLATABLE SPACE STRUCTURES

## I. Background and Statement of Problem

### 1.1 Background

On 20 May 1996, the astronauts of the Space Shuttle Endeavor, STS-77, used the remote manipulator system to deploy Spartan 207/Inflatable Antenna Experiment (IAE). Shortly thereafter, the Spartan bus initiated the deployment of an inflatable antenna developed by L'Garde Inc and NASA's Jet Propulsion Laboratory (JPL) under NASA's In-Space Technology Experimental Program. The antenna consisted of a 14-meter diameter reflector surface and transparent canopy, joined at the aperture by a strong, flexible torus. The torus was connected to the spacecraft bus by three 28-meter inflatable struts. The deployed antenna is shown in Figure 1.1. Despite experiencing unexpected dynamics during initial ejection and inflation, the antenna attained the correct final shape in just ten minutes. After recording data and video images of the antenna during one complete orbit, the Spartan spacecraft ejected the inflatable structure. The shuttle crew retrieved the bus and returned its data and images to Earth. Although the lenticular structure itself failed to completely inflate, the mission was considered a great success. It validated the deployment and dimensional stability of a large inflatable antenna in an operational orbit. This successful mission sparked renewed interest and research into the use of inflatable structures for space missions.

Inflatable structures have several distinct advantages over alternative deployment methods. First and foremost is lower cost. Inflatable structures weigh 50% less and can be stowed in 25% of the volume of the best competing mechanical
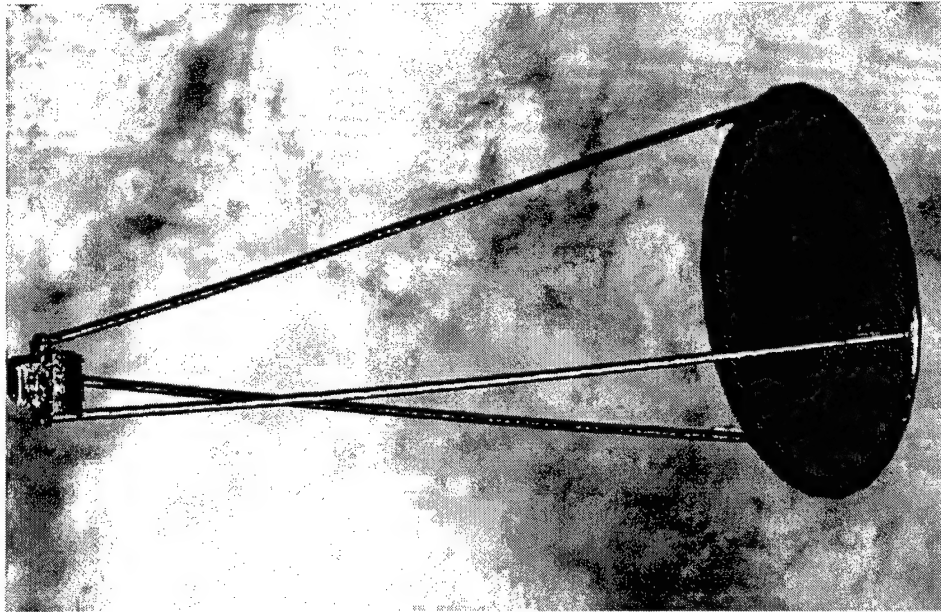
Figure 1.1    Inflatable Antenna Experiment

systems [3]. This permits the selection of a smaller launch vehicle or reduction in the number of vehicles for a mission which previously required multiple launches and subsequent on-site assembly. The resultant savings is tens or even hundreds of millions of dollars. Furthermore, inflatable structures themselves are less expensive than their mechanically-deploying counterparts due to their simple engineering and low production costs. The basic elements of an inflatable structure consist of flat sheets of material, seams and adhesives, launch packaging and an inflation system [3]. The materials used are not exotic, and the tooling required is relatively inexpensive. JPL estimates that the cost of an inflatable structure, such as an antenna, will be an order of magnitude less than its mechanical equivalent [7].

Cost is not the only advantage. Other advantages of inflatable structures include strength, reliable deployment and sound thermal response. Inflatables are inherently strong due to their ability to absorb loads over a large surface area. In contrast, mechanical structures usually have loads concentrated in a few points, which must be strengthened and therefore, made heavier. Inflatables also deploy more reliably than mechanical systems. When properly designed, the only failure point for

deployment of the structure is the initiation of gas release. Furthermore, the deployment sequence is self-correcting. If the structure starts to hang up, the pressure from the inflation system increases until the obstruction is overcome. The importance of this reliability was illustrated by the recent problems with the mechanically deployed solar arrays on the International Space Station. Improper deployment left the array in a "crumpled" configuration incapable of delivering the design power load. A costly and potentially dangerous spacewalk was required to correct the fault. In higher orbits, manual repair is not even an option. Finally, inflatable structures can be engineered to display favorable thermal response. The large, opposing surface areas allow the use of radiation exchange to reduce thermal gradients. In fact, by carefully selecting the internal and external optical properties, engineers can limit the temperature difference between sunlit and shaded elements to 10 K or less [3].

The use of inflatable structures in space is not a new concept. In the early 1960's, NASA orbited several inflatables, including the passive communications satellites, Echo I and II. Although inflatable technology was in its infancy, the limited capacity of the launch vehicles in that era demanded that the missions be performed with inflatables, or not at all. As more powerful launch systems were brought online, contractors focused on developing mechanically-deployed structures. America was engaged in a space-race with the Soviet Union which meant that the accepted solution was the one which could be most easily done, not necessarily the optimum one [3]. The industry was familiar with mechanical systems and possessed the skilled engineers and analytical tools needed to develop and test these systems. In addition, experts were concerned with the survivability of inflatable structures in space. The meteoroid threat was not well defined and generally overestimated. As a result, effort was focused on the development of inflate-then-rigidize systems. These systems are more difficult to manufacture and are heavier than inflate-only systems. This further encouraged emphasis on mechanical systems. Eventually, these systems dominated the industry. This had a snowball effect, leading to the use of mechanical structures

even on missions where inflatable structures would provide distinct advantages. As Michael Dornheim puts it, inflatable technology suffered from the chicken-or-egg problem [7]. Program managers would not select inflatable structures because they were not space-qualified. Yet, the reason they were not space-qualified is that they were not being used.

However, times change. The lack of any significant increase in lift capability in recent years has increased the focus on developing lighter-weight structures with minimal stowed volume. Also, shrinking budgets have forced the space community to look for more efficient and economical methods of performing space missions. The meteoroid threat is better defined as well. The conservative estimates for meteoroid flux used in the past were three orders of magnitude too high [3]. Even if micro-meteoroid damage occurs, these structures are inflated at such low pressures that enough makeup gas for a ten-year lifetime can easily be carried. So now, the space industry is looking again at using inflatable structures to reduce costs and improve capabilities. Companies such as L'Garde, Thoikol, Contraves and Aerospace Recovery Systems are developing large inflatable structures technology for a variety of missions. Antennas similar to the IAE will be used in JPL's Advanced Radio Interferometry between Space and Earth (ARISE) mission. This concept uses orbiting 30-foot inflatable antennas in conjunction with ground antennas to synthesize a radio frequency (RF) interferometer with a baseline longer than the earth's diameter. This allows high angular resolution of distant RF sources [23]. Inflatable solar arrays are another promising application. L'Garde has built a prototype which produces 60 watts/kg. Experts at JPL believe that this can easily be pushed to 90 watts/kg, which doubles the state of the art for mechanical arrays (44 watts/kg on Deep Space 1)[7]. L'Garde is also working with ILC Dover to develop a 32 by 14 meter inflatable sun shield for the Next Generation Space Telescope (NGST). Other potential missions for inflatables include lightweight solar sails (Deep Space 5), solar concentrators, optical telescope mirrors, supports for synthetic aperture radars, and

pressurized habitat modules for the International Space Station [7]. The possibilities are endless. Not only do the inflatable structures reduce production and spacelift costs, they enable missions that were previously thought impossible.

The Air Force will also benefit from inflatable technology. Desert Storm demonstrated how integrating weather, navigation, missile warning and communications data from space-based assets can drastically improve the effectiveness of ground, sea and air forces. Effectiveness is further increased by providing theater commanders with near real-time imagery of the battlefield. Current imaging systems are limited in resolution or orbital altitude by the size of the optics required and the lift capacity of the launch vehicle. Use of inflatable optics will permit the launch and deployment of imagers with much larger apertures. These sensors will provide better resolution from higher altitudes, extending coverage times and making the platforms more survivable. Difficulty arises due to the tight surface precision required of optical lenses. However, improved methods of rigidizing inflatables and increasing precision with which the desired shape is attained promise the development of inflatable lenses in the near future. Inflatable structures are the future of space operations for missions requiring large apertures.

## 1.2 Problem Statement

With the space industry adopting inflatables for a host of space missions, better tools are needed to analyze the behavior of these large, flexible structures under the mechanical and thermal loads encountered in space. Since RF and optical applications require tight surface precision, prediction and control of structure behavior is critical. Analytic programs used in the past, such as NASTRAN, were designed to analyze structures with small deflections [3]. However, inflatable structures are load adaptive structures, accommodating various forces by changing geometry. Models are being developed to better analyze inflatables. One promising method is finite element analysis. This is a process of dividing a large structure into many smaller units

(finite elements) joined at common points called nodes. Instead of trying to solve for the behavior of the entire body at once, displacement equations are developed for each element and then combined to find the solution for the whole structure. This process makes it possible to find a solution for problems involving complex geometries, loading, and material properties [16].

Solving the finite element model requires knowledge of the external loads. The applied forces, pressures and heat flux must be determined for each element and applied at the correct nodes. For space missions, this requires calculation of the loads encountered in the space environment. That is the purpose of this research: to determine and model the critical loads encountered by a large inflatable structure in orbit.

## 1.3 Scope

A MATLAB program will be developed to model the forces encountered by a large inflatable structure as it propagates through a variety of orbits. Chapter 2 will present background theory and a review of current literature concerning disturbances experienced by orbiting spacecraft. The model will focus on environmental forces, ignoring any loads generated internally from the satellite. Rough calculations will be performed to determine the order of magnitude of each disturbance, allowing identification of the critical loads. Once the critical loads are determined, a MATLAB algorithm will be developed to more accurately determine the loads on each element of an orbiting structure. Chapter 3 will describe the methodology used in developing this program. Chapter 4 will present the final disturbance model along with required inputs and sample output for a basic structure. Finally, chapter 5 will provide a summary of conclusions drawn, lessons learned and recommendations for further research.

## II. Literature Review and Preliminary Calculations

### 2.1 Critical Load Identification

A key step in the design of any structure is the identification and quantification of the critical loads. The critical or primary loads are the most significant forces acting on a structure which will in turn have the most pronounced effect on performance. According to Hedgepath [11], the successful performance of any structure demands accurate determination of the critical loads and related design criteria. Furthermore, mission feasibility studies and life cycle cost projections require proper identification of these critical loads and design requirements. This is especially true for missions where large structures dominate the spacecraft design. Therefore, determination of critical loads will be the first step in modeling inflatable structures. A literature review of the loads experienced by a spacecraft will be presented in this chapter, followed by basic calculations for inflatable antenna experiment shown in Figure 1.1. Comparison of the relative magnitudes of these forces will facilitate identification of the critical loads.

Accurate identification of critical loads requires analysis of each phase in the spacecraft life cycle. These phases include prelaunch, launch, inter-orbit boost, deployment, and space operations [11]. Prelaunch activities include the development, fabrication and testing of the spacecraft components as well as assembly and transport to the launch pad. Launch and inter-orbit boost phases encompass all the needed maneuvers to place the satellite in its operational orbit. Deployment includes the pressurization of inflatable structures and/or the deployment of mechanical structures. Space operations covers the daily operations as well as the end-of-life disposal of the satellite. In the past, critical loads arose from the launch environment and inter-orbit boost phase. Strength was the primary design criteria in order for the satellite structure to survive the extreme forces experienced during launch. The smaller loads encountered on orbit during the operations phase were of sec-

ondary concern. However, deployable structures such as inflatables will be in their packaged configuration during the launch and boost phases. They will not be deployed until reaching the operational orbit. Therefore, the loads encountered in the space environment will become the critical loads. Since the loads experienced in space are relatively small, strength is no longer a critical design requirement. For many applications of inflatable structures such as antennas, solar collectors or optics, dimensional accuracy throughout the operational life will be the primary design requirement. This demands structures with adequate stiffness to resist deformations. However, in order to fully reap the benefits of inflatable structures, they must be low volume and low weight. With current launch systems, volume restrictions are generally more difficult to meet than weight restrictions [11]. Therefore, low volume of the the stowed configurations is also a critical design criteria. Successful performance of an inflatable space structure will hinge on satisfying each of these critical design requirements.

In order to strike an optimal balance between low volume and adequate stiffness in the structure design, the critical loads during the operational phase must be identified and quantified. Loads encountered on orbit may be generated internal to the spacecraft or result from interaction between the satellite and the space environment. The most significant on-board disturbances include heating from the operation of spacecraft subsystems and torques generated by the attitude control mechanism [9]. The impact of these internally generated loads can be quite severe. For example, operation of the attitude control subsystem can cause high frequency mechanical disturbances which may initiate a structural response at the natural frequencies of the satellite, resulting in unstable oscillations. However, the exact magnitude and distribution of these internal loads is dependent on many factors such as the type of attitude control system used, the amount of power used by spacecraft subsystems, thermal shielding and insulation of various components, location of heat sources, location of thrusters, length and magnitude of thruster firings and so on. Obvi-

ously, the loading profile of these internal disturbances can vary significantly from one spacecraft design to another. The modeling of these internal loads is beyond the scope of this research effort. Instead, attention will be focused on loads arising from interaction between the spacecraft and its environment.

There are many different aspects of the space environment which impose loads on the spacecraft [1]. The earth's gravitational field creates torques on large satellites. Smaller torques can arise from interaction with the moon or sun's gravitational field. The earth's magnetic field can interact with the spacecraft's magnetic moment to generate torques. Spacecraft traveling through the upper portion of the earth's atmosphere experience drag forces which oppose velocity, degrading the satellite orbit. Impact with meteoroids or man-made debris can damage the satellite and generate torques. Rapidly changing thermal fluxes due to solar radiation and earth infrared emission can induce torques in large space structures. Solar radiation and solar wind create pressure forces on large satellite surfaces which can perturb the orbit and affect satellite attitude. The load imposed by these forces vary depending on satellite configuration, orbital altitude and inclination, solar activity, time of day, season and many other factors. However, a reasonably accurate model can be generated by making some simplifying assumptions and modeling only the critical forces.

Several of the disturbances mentioned above are of secondary concern. For satellites at or below geosynchronous altitudes, the gravitational effects of the moon and sun are negligible [9]. Similarly, the pressure generated by solar wind is several orders of magnitude lower than solar radiation pressure and can likewise be neglected [9]. Impact with meteoroids can generate critical loads, but these impacts are very rare and difficult to predict. The resulting torques from impact are even more challenging to predict. This leaves gravity gradient, solar radiation pressure, aerodynamic drag, magnetic torques and thermally induced torques as possible critical forces.

Garrett theorizes that the most critical loads are solar radiation pressure, gravity gradient, drag, and thermal radiation [9]. Chobotov states that the environmental effects of greatest concern in satellite design are solar radiation pressure, gravity gradient, aerodynamic drag and magnetic torque [4]. Furthermore, the significance of each load varies depending on the orbit. Figure 2.1 below, taken from Chobotov, shows the typical environmentally induced torques as a function of orbital altitude [4]. Note that solar radiation pressure is constant, while aerodynamic drag drops off rapidly with increasing altitude. Gravity gradient and magnetic torque follow similar curves, dropping off more rapidly as altitude increases. In general, gravity gradient and magnetic torques are considered significant below 1000 kilometers(km) altitude. Drag is significant below 500 km and decays rapidly, becoming negligible above 1000 km. Solar radiation pressure becomes significant above 1000 km when gravity, magnetics and drag have diminished. Hence, the critical loads are a function of altitude. The critical loads for a low-earth orbiting spacecraft will differ from those of a geosynchronous satellite.
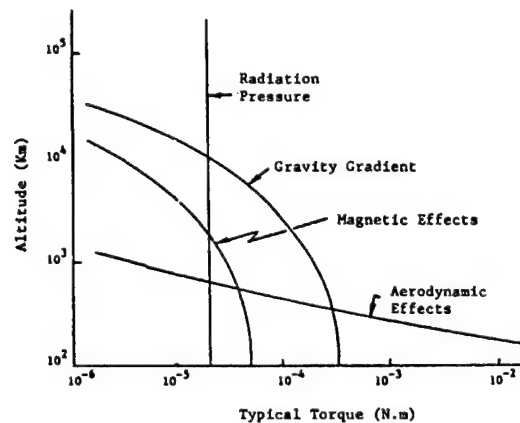


Figure 2.1    Torque vs. Altitude for Small Satellites [4]

Figure 2.1 does not provide a complete picture. The torques are shown as a function of altitude only. These torques are typical values for small satellites [4]. However, the induced torques are also dependent on the spacecraft geometry and

attitude. The magnitude of the torques encountered by a large structure could vary significantly from those imposed on a small satellite. Most inflatables structures will be large structures. Also, Figure 2.1 does not show the relative magnitude of thermally induced torques. In many orbits, heat flux into the satellite varies slowly over time, creating negligible torques. However, when satellites enter or exit eclipse, rapidly changing thermal conditions can induce much larger torques which can significant impact on satellite dynamics. This effect must be taken into account when modeling critical forces. Therefore, a more precise approximation of these disturbances for large structures must be developed which accounts for structure geometry and attitude, as well as thermal torques. Using the IAE as the baseline model, a plot similar to Figure 2.1 will be generated for large spacecraft. From this plot, the critical loads will be identified.

## 2.2 IAE Model

As previously mentioned, the IAE is an inflatable parabolic antenna connected to a Spartan bus. Reference [8] contains a detailed description of the antenna construction and dimensions. The lenticular structure consists of two circular canopies, one aluminized (reflector surface) and one transparent. Each of the 14 meter diameter canopies is formed by joining together pie-shaped gores of 6.35 micron mylar film. The canopies are joined at the edge by a strong, but flexible torus. The rim support provided by the torus causes the canopies to attain the desired parabolic shape when fully inflated. The torus connects to three 28-meter long cylindrical struts which extend the reflector out from the satellite bus. The struts and torus are made from 0.3 millimeter (mm) neoprene coated Kevlar. The cross section diameter of the torus is 0.61 meters (m) and the diameter of the struts is 0.36 m. The antenna structure has a total mass of 60 kg and is connected to the Spartan bus which has a mass of 900 kg [25]. Table 2.1 contains a summary of the material properties. Properties

marked by an asterisk are assumed values. Other values have been slightly altered to protect L'Garde Inc. proprietary information.

Table 2.1    IAE Material Properties

| Property | Mylar | Kevlar |
|---|---|---|
| Density($\rho$) | $1400\ kg/m^3$ | $1370\ kg/m^3$ |
| Reflection coefficient($\beta$) | 0.9* | 0.6 |
| Thermal expansion coefficient($cte$) | $5.4e-7/K$* | $5e-7/K$ |
| Thermal conductivity($k$) | $216.3\ W/m/K*$ | $216.3\ W/m/K*$ |
| Solar absorptivity($\alpha$) | 0.1* | 0.4 |
| Thermal emissivity($\epsilon$) | 0.7* | 0.9 |
| Specific heat($c_p$) | $500\ J/kg/K*$ | $500\ J/kg/K*$ |

The IAE can be modeled as a composite of six basic shapes. The reflector and transparent canopy can be represented together as a laminar disk. The three struts are hollow cylinders and the spartan bus a rectangular box. The torus, of course, is just a torus. For simplicity sake, the three struts are assumed to converge at the Spartan's center of mass. A body-fixed reference frame can be defined with the orgin at the Spartan center of mass and the $b_1$ axis pointed through the center of the reflector assembly as shown in Figure 2.2. The struts attach to the torus at evenly spaced 120 degree intervals. Using standard formulas [19] and applying some
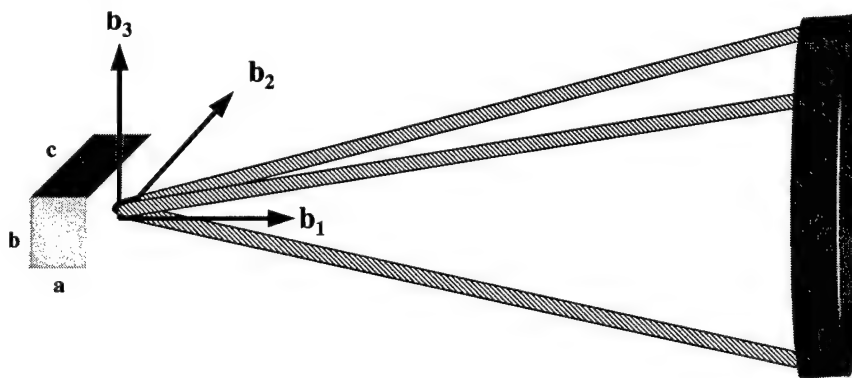


Figure 2.2    IAE Model with Body-Fixed Axes

basic geometry yields the center of mass location and moment of inertia about the center of mass for each individual component. The spacecraft center of mass is then

determined by the center of mass equation for a system of particles.

$$r^{cm} = \frac{1}{M} \sum_{i=1}^{n} m_i r_i \qquad (2.1)$$

where

$$
\begin{aligned}
M &= \text{total body mass} \\
m_i &= \text{component mass} \\
r_i &= \text{body fixed coordinates of component center of mass}
\end{aligned}
$$

Treating each component as an equivalent point mass and applying (2.1) yields the spacecraft center of mass in body fixed coordinates.

$$r^{cm} = \begin{bmatrix} 1.15m & 0m & 0m \end{bmatrix} \{\hat{b}\} \qquad (2.2)$$

Likewise, the spacecraft inertia matrix is found by summing the individual inertia matrices. In order to sum inertia matrices, the matrices must be in the same reference frame and about the same point. The body fixed inertia matrix for each component about the spacecraft center of mass is calculated using the parallel axis theorem.

$$I^c = I^{ci} + m^i * \begin{bmatrix} \Delta y^2 + \Delta z^2 & -\Delta y \Delta x & -\Delta z \Delta x \\ -\Delta x \Delta y & \Delta x^2 + \Delta z^2 & -\Delta z \Delta y \\ -\Delta x \Delta z & -\Delta y \Delta z & \Delta x^2 + \Delta y^2 \end{bmatrix} \qquad (2.3)$$

$I^c$ is the component's inertia about the spacecraft center of mass, and $I^{ci}$ is the component's inertia about its own center of mass. $\Delta x$, $\Delta y$ and $\Delta z$ represent the distance from the component center of mass to the spacecraft center of mass along the $\hat{b}_1$, $\hat{b}_2$ and $\hat{b}_3$ axes respectively. Summing the component inertia matrices yields
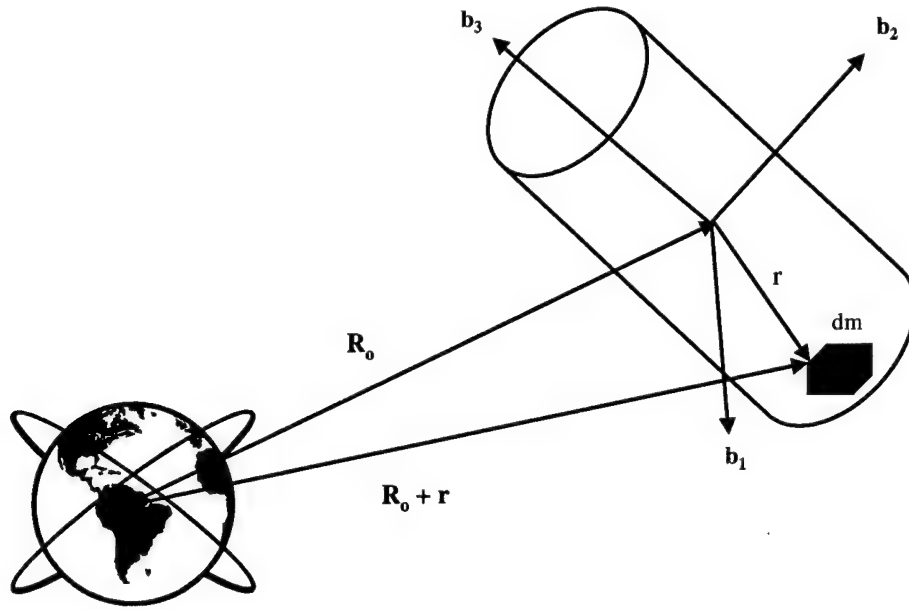
Figure 2.3    Gravity Gradient Model

the spacecraft inertia matrix.

$$I^c = \begin{bmatrix} 3432 & 0 & 0 \\ 0 & 25656 & 0 \\ 0 & 0 & 26324 \end{bmatrix} kgm^2 \qquad (2.4)$$

Detailed calculations for the spacecraft center of mass and inertia matrix are listed in Appendix C.

## 2.3   Gravity Gradient

One possibly significant disturbance on large satellites in low-earth orbit (LEO) is gravity gradient. Gravity gradient torques arise due to the varying acceleration of gravity across the distributed mass of a large space structure. Consider the large satellite shown in Figure 2.3. The gravitational torque on the satellite is determined by

$$M = \int (\vec{r} \, X \, a_g) \, dm \qquad (2.5)$$

The gravitational acceleration $a_g$ is

$$a_g = -\mu \frac{\vec{R}_o + \vec{r}}{|\vec{R}_o + \vec{r}|^3} \qquad (2.6)$$

where the gravitational parameter $\mu$ is the product of the earth's mass and the gravitational constant, $/vecR_o$ is the radius from the center of the earth to the spacecraft center of mass, and $\vec{r}$ is the radius from the spacecraft center of mass to the element of interest. Now let $X, Y$, and $Z$ be the components of the vector $\vec{R}_o$ in the body fixed frame and let $x, y$ and $z$ be the components of $\vec{r}$. Substituting (2.6) into (2.5) and expanding the cross product in the numerator yields

$$\vec{r} \, X \, (\vec{R}_o + \vec{r}) = (yZ - zY)\hat{b}_1 + (zX - xZ)\hat{b}_2 + (xY - yX)\hat{b}_3 \qquad (2.7)$$

Assuming that the radius of the orbit is much larger than the size of the spacecraft, the denominator of (2.6) can be expanded using the binomial expansion theorem. Dropping all higher order terms gives

$$|\vec{R}_o + \vec{r}|^{-3} \approx R_o^{-3} [1 - \frac{3(Xx + Yy + Zz)}{R_o^2}] \qquad (2.8)$$

Substituting (2.8) and (2.7) into (2.5) provides the following expression for the $b_1$ component of torque.

$$
\begin{aligned}
M_1 \;=\; & -\frac{\mu}{R_o^3}[Z \int y\,dm \;-\; Y \int z\,dm \\
& -\frac{3XZ}{R_o^2} \int xy\,dm \;-\; \frac{3YZ}{R_o^2} \int (y^2 - z^2)\,dm \\
& -\frac{3Z^2}{R_o^2} \int yz\,dm \;+\; \frac{3XY}{R_o^2} \int xz\,dm \;+\; \frac{3Y^2}{R_o^2} \int zy\,dm]
\end{aligned} \qquad (2.9)
$$

Assuming that the orgin of the body frame is located at the center of mass, the first two integral terms in (2.9) go to zero. Also, let the body frame be the principle axis

2-9

set. All products of inertia go to zero leaving

$$
\begin{aligned}
M_1 &= -\frac{3\mu YZ}{R_o^5}[-\int y^2 dm + \int z^2 dm] \\
&= \frac{3\mu YZ}{R_o^5}\int [x^2 + y^2 - (x^2 + z^2)]dm \\
&= \frac{3\mu YZ}{R_o^5}(I_{33} - I_{22})
\end{aligned}
\tag{2.10}
$$

The other two components of torque can also be expanded and simplified to obtain similar expressions.

$$
M_2 = \frac{3\mu XZ}{R_o^5}(I_{11} - I_{33})
\tag{2.11}
$$

$$
M_3 = \frac{3\mu XY}{R_o^5}(I_{22} - I_{11})
\tag{2.12}
$$

Equations (2.10)–(2.12) can be used to determine the gravity gradient torque on a space structure given its orbital position. However, recall that $X$,$Y$ and $Z$, the components of the radius vector, are expressed in the satellite body frame, and therefore change with time. In order to analyze the dynamics of the satellite using Euler's equations, the position coordinates should be expressed in terms of orientation angles. Assume a circular orbit and define the local orbit frame, $\{\hat{a}\}$, with its orgin at the satellite center of mass, $\hat{a}_1$ pointing along the radius vector, $\hat{a}_2$ along the velocity vector and $\hat{a}_3$ along the orbit normal. In this frame, $\vec{R}_o$ is simply $R_o \hat{a}_1$. Now let the orientation of $\{\hat{b}\}$ with respect to $\{\hat{a}\}$ be expressed as an Euler 1-2-3 rotation as shown in Figure 2.4. The transformation matrix from $\hat{a}$ to $\hat{b}$ is easily calculated.

$$
C^{ab} = \begin{bmatrix}
c\theta_3 c\theta_2 & c\theta_3 s\theta_2 s\theta_1 + s\theta_3 c\theta_1 & -c\theta_3 s\theta_2 c\theta_1 + s\theta_3 s\theta_1 \\
-s\theta_3 c\theta_2 & -s\theta_3 s\theta_2 s\theta_1 + c\theta_3 c\theta_1 & s\theta_3 s\theta_2 c\theta_1 + c\theta_3 s\theta_1 \\
s\theta_2 & -s\theta_1 c\theta_2 & c\theta_2 c\theta_1
\end{bmatrix}
\tag{2.13}
$$
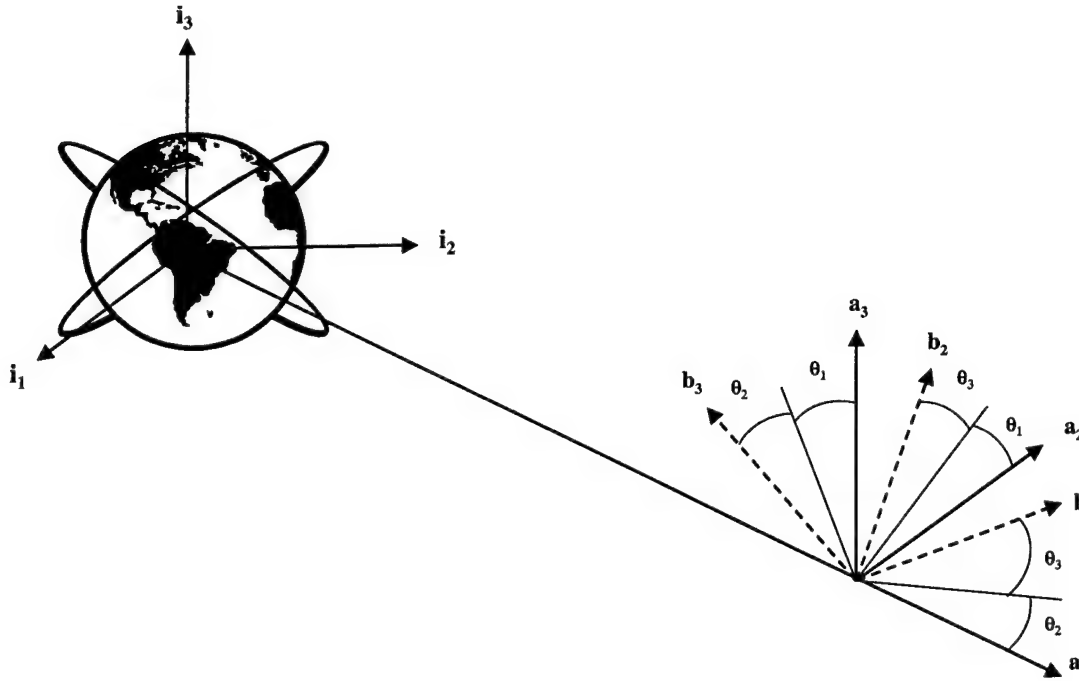
Figure 2.4    Orbit Frame Geometry

where $c$ and $s$ are shorthand for the cosine and sine functions. This transformation can be used to express $\vec{R}$ in the $\vec{b}$ frame.

$$
{}^{b}\vec{R}_o = C^{ab}{}^{a}\vec{R}_o = C^{ab} \begin{pmatrix} R_o \\ 0 \\ 0 \end{pmatrix}
\tag{2.14}
$$

Recalling that $X$, $Y$ and $Z$ are the components of ${}^{b}\vec{R}_o$ yields

$$
\begin{aligned}
X &= R_o \cos\theta_3 \cos\theta_2 \\
Y &= -R_o \sin\theta_3 \cos\theta_2 \\
Z &= R_o \sin\theta_2
\end{aligned}
\tag{2.15}
$$

2-11

Substituting (2.15) into (2.10)–(2.12) provides the following expression for gravity gradient torques.

$$M_1 = \frac{-3\mu \sin\theta_3 \sin\theta_2 \cos\theta_2}{R_o^3}(I_{33} - I_{22})$$

$$M_2 = \frac{3\mu \cos\theta_3 \cos\theta_2 \sin\theta_2}{R_o^3}(I_{11} - I_{33})$$

$$M_3 = \frac{-3\mu \cos\theta_3 \sin\theta_3(\cos\theta_2)^2}{R_o^3}(I_{22} - I_{11}) \qquad (2.16)$$

Visual examination of (2.16) leads to several conclusions about gravity gradient torque. First, the torques are a function of the difference between the satellite's principle moments of inertia. Therefore, gravity gradient torques are most significant on large, asymmetric bodies. Secondly, the torques are inversely proportional to the orbital radius cubed. This means that while gravity gradient is strong for LEO satellites, it drops off quickly with increasing orbital altitude. Finally, if all of the orientation angles are equal to zero, there is no torque due to gravity gradient. These equilibrium points require that one principle axis be aligned with the orbit normal and that the satellite spin about that axis at a rate equal to the mean motion of the orbit. Also, a second principle axis must be aligned with the radius vector. When these conditions are met, gravity gradient torques are zero. Six such equilibrium points exist since any one of the three principle axes can be aligned with the orbit normal and each of these alignments has two options as to which axis to align with nadir. However, solving Euler's equations for small perturbations from these equilibrium points shows that only one orientation is stable [30]. This stable orientation occurs when the satellite's major inertia axis is aligned with the orbit normal and the minor inertia axis is aligned with the orbit radius vector. Any perturbation from this orientation creates a gravity gradient torque which acts as a restoring force. The more asymmetric the body (larger difference between major and minor moments of inertia) and the larger the perturbation, the greater the restoring torque. This torque can impose a significant burden on the control system of a large

2-12

asymmetric satellite if the body is maintained in orientation other than its stable equilibrium.

The stable configuration for the IAE is with the $\hat{b}_1$ axis along the radius vector and the $\hat{b}_3$ axis along the orbit normal. This would most likely be the nominal attitude with the antenna directed along nadir. In this orientation, gravity gradient torque would be zero. Assume now that the antenna is slewed to focus off-nadir. A gravity gradient torque will arise, acting as a restoring force towards the stable attitude. In order to get a true feel for the potential impact of each torque, calculations will be made for the worst case orientation (greatest induced torque). Examination of equation(2.16) reveals that the worst case for gravity gradient occurs when the IAE is rotated 45 degrees about the $\hat{b}_2$ axis. Limiting the analysis to just the single axis rotation and substituting IAE inertia properties into equation (2.16) provides the following relationship between torque and orbital altitude,

$$
\begin{aligned}
M_1 &= 0 \\
M_3 &= 0 \\
M_2 &= \frac{1.363 * 10^{19}}{(R_e + h)^3} Nm
\end{aligned}
\tag{2.17}
$$

where $R_e$ is the radius of the earth and h is the orbital altitude. Both $R_e$ and h are expressed in meters. Table 2.2 shows the gravity gradient torques on the IAE at various altitudes.

Table 2.2    Gravity Gradient Torques

| Altitude(km) | Torque(Nm) |
|---|---|
| 300 | $4.58 * 10^{-2}$ |
| 500 | $4.18 * 10^{-2}$ |
| 1000 | $3.39 * 10^{-2}$ |
| 2000 | $2.32 * 10^{-2}$ |
| 5000 | $9.25 * 10^{-3}$ |
| 26610 | $3.80 * 10^{-4}$ |
| 42240 | $1.19 * 10^{-4}$ |

## 2.4 Magnetic Torques

The earth has a magnetic field surrounding it which can interact with orbiting spacecraft. It is widely believed that motion in the liquid part of the earth's core generates currents which subsequently induce the magnetic field [26]. A dipole field with its magnetic axis offset 11.5 degrees from the rotational axis provides a reasonably accurate representation of the earth's magnetic field. This model is illustrated in Figure 2.5. Anomalies do occur in the field due to concentrations of ferrous metals in various locations on the earth's surface. Also, a small portion of the field (less than 0.1%) is generated by the motion of charged particles in the ionosphere [26]. These currents vary based on the intensity of solar activity and the resulting solar wind. However, over 90% of the earth's field is accounted for by the homogeneous dipole field model [4]. This will provide an adequate degree of accuracy to develop rough order of magnitude estimates for the magnetic torques imposed upon the IAE. The dipole field $\vec{B}_o$ is determined by the gradient of the earth's dipole potential.

$$\vec{B}_o = -\nabla\phi \qquad (2.18)$$

where

$$\phi = \frac{\mu_e}{r^2}\cos\theta$$

$$\nabla = \hat{e}_r\frac{\partial}{\partial r} + \hat{e}_\theta\frac{1}{r}\frac{\partial}{\partial\theta}$$

$r$ = distance from the center of the earth

$\theta$ = angle from the magnetic axis to the radius vector

The field is expressed in terms of the polar coordinates shown in Figure 2.5. The magnitude of the earth's magnetic moment vector along the dipole axis, $\mu_e$, is approximately $8.05 * 10^{25} gauss - cm^3$. Taking the gradient results in the following
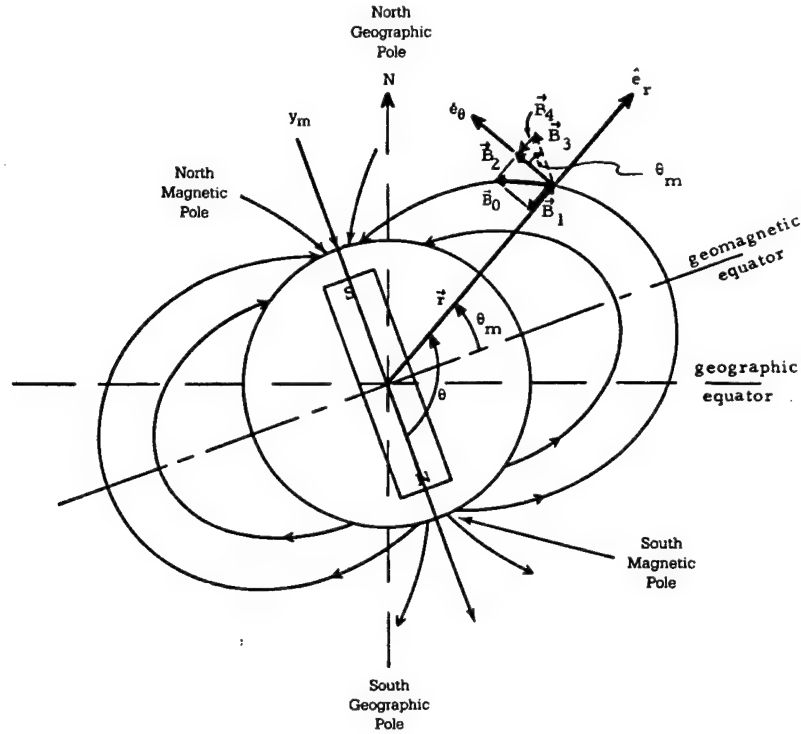
Figure 2.5    Earth's Magnetic Field [4]

expression for the dipole field

$$
\begin{aligned}
\vec{B}_o &= \frac{\mu_e}{r^3}(2\cos\theta\hat{e}_r + \sin\theta\hat{e}_\theta) \\
&= \frac{\mu_e}{r^3}(-2\sin\theta_m\hat{e}_r + \cos\theta_m\hat{e}_\theta)
\end{aligned}
\tag{2.19}
$$

where $\theta_m$ is the magnetic latitude measured from the magnetic equator as shown in Figure 2.5. The magnitude of the field simplifies to

$$
B_o = \frac{\mu_e}{r^3}[1 + 3(\sin\theta_m)^2]^{\frac{1}{2}}
\tag{2.20}
$$

Equation 2.20 shows that the strength of the magnetic field encountered by a satellite is dependent on the magnetic latitude. The field is strongest at the magnetic poles ($\theta_m = 90\,\deg$) and weakest above the magnetic equator ($\theta_m = 0\,\deg$). Also, the strength of the field is inversely proportional to the radius cubed, just like gravity

2-15

gradient. Therefore, while the torques induced may be significant for LEO satellites, they will drop off quickly with increasing altitude.

An orbiting spacecraft can posses a residual magnetic field. This field can be generated by temporarily magnetized materials on the spacecraft, eddy currents of charged particles around a spinning satellite, or current loops within the spacecraft [4]. This residual field can interact with the earth's magnetic field to generate a torque on the spacecraft in the following manner.

$$\vec{T} = \vec{M} X \vec{B_o} \tag{2.21}$$

where $\vec{T}$ is the magnetic torque and $\vec{M}$ is the magnetic moment of the spacecraft. The magnitude of the torque is

$$T = M B_o \sin \alpha \tag{2.22}$$

where $\alpha$ is the angle between the two vectors. The magnitude of the earth's field is given by equation (2.20).

Suppose the IAE has an unshielded current loop in one of the struts extending from the Spartan bus to an actuator on the torus. In the case of a current loop, the magnitude of the magnetic moment is simply the product of the current, $i_c$, and the area, A, enclosed by the loop [30].

$$M = i_c A \tag{2.23}$$

The greatest torque will occur when the satellite is above a magnetic pole where the earth's field is strongest. In this case, $\theta_m = 90 \deg$ and equation (2.20) reduces to

$$B_o = \frac{2\mu_e}{r^3} \tag{2.24}$$

Substituting (2.23) and (2.24) into (2.22) yields

$$T = \frac{2i_c A \mu_e}{r^3} \sin \alpha \qquad (2.25)$$

Recall that the IAE strut is 28m long. Assume that the wires are separated by 1 centimeter and carry a current of 10 ampheres. While typical load currents may be lower [20], 10 amps was chosen to provide a conservative, worst-case estimate. Also, assume that the magnetic moment is perpendicular to the earth's local field ($\alpha = 90 \deg$). Plugging these values into equation (2.25) provides the following expression for magnetic torque as a function of altitude.

$$T = \frac{4.508 * 10^{16}}{(R_e + h)^3} Nm \qquad (2.26)$$

Table 2.3 lists the magnitude of the magnetic torque on the IAE at various altitudes.

Table 2.3    Magnetic Torques

| Altitude(km) | Torque(Nm) |
|---|---|
| 300 | $1.50 * 10^{-4}$ |
| 500 | $1.38 * 10^{-4}$ |
| 1000 | $1.12 * 10^{-4}$ |
| 2000 | $7.66 * 10^{-5}$ |
| 5000 | $3.00 * 10^{-5}$ |
| 26610 | $1.26 * 10^{-6}$ |
| 42240 | $3.92 * 10^{-7}$ |

## 2.5   Solar Radiation Pressure

Solar radiation pressure is another potential source of torque on an orbiting space structure. Radiation pressure occurs due to the particle nature of light. Photons can be treated analytically as particles with mass governed by $E = mc^2$. When these particles collide with a surface, a transfer of momentum occurs. The resultant

force on the surface is equal to the time rate of change of momentum($p$).

$$F = \frac{dp}{dt} \tag{2.27}$$

For a collimated beam of light, the momentum per unit volume is given by $H/c^2$ where $c$ is the speed of light and $H$ is the power per unit area [9]. The amount of momentum transferred to an absorbing surface of area $A$ during the time interval $\triangle t$ is given by

$$\triangle p = \frac{H}{c^2} * c * A * \triangle t = \frac{H}{c} A \triangle t \tag{2.28}$$

The force imposed on the surface by the radiation is just

$$f_r = \frac{\triangle p}{\triangle t} = \frac{H}{c} A \tag{2.29}$$

where the term $H/c$ is the radiation pressure. The solar flux, $H$, is dependent on the distance from the source. When trying to determine solar radiation pressure on earth-orbiting satellite, H is taken to be the solar constant, $1353W/m^2$, which is the mean solar flux at a distance of one astronomical unit (AU) from the sun. The solar flux at the edge of the earth's atmosphere actually varies from $1300W/m^2$ at apohelion to $1400W/m^2$ at perihelion [4]. However, the solar constant provides a reasonable approximation to be used on solar radiation pressure calculations.

The term $H/c$ calculates the solar radiation pressure for a perfect absorber at normal incidence which is equal to $4.5 * 10^{-6} N/m^2$. Conservation of momentum dictates that the pressure imposed on a perfect reflector would be twice as great since the photons are not just stopped, but then sent back the opposite direction with equal velocity. However, most satellite surfaces are not perfect absorbers or reflectors; they are somewhere in-between. Furthermore, there are different types of reflection as shown in Figure 2.6. On a specular surface the reflection is directional, meaning that the angle of incidence is equal to the angle of reflection. For diffuse

surfaces, radiation is reflected in all directions. According to Lambert's cosine law, the intensity of the reflected radiation in any direction from a diffuse surface is proportional to the cosine of the angle of reflection. Of course, most surfaces are not perfectly specular or diffuse. As a result, reflection from these surfaces will be partially diffuse, a combination of specular and diffuse reflection. Also, earth-orbiting satellites experience pressure from earth-reflected solar radiation and earth-emitted thermal radiation. However, previous analytical work has proven that pressure from direct solar radiation is generally more than an order of magnitude higher than the other terms and thus dominates the interaction[9].
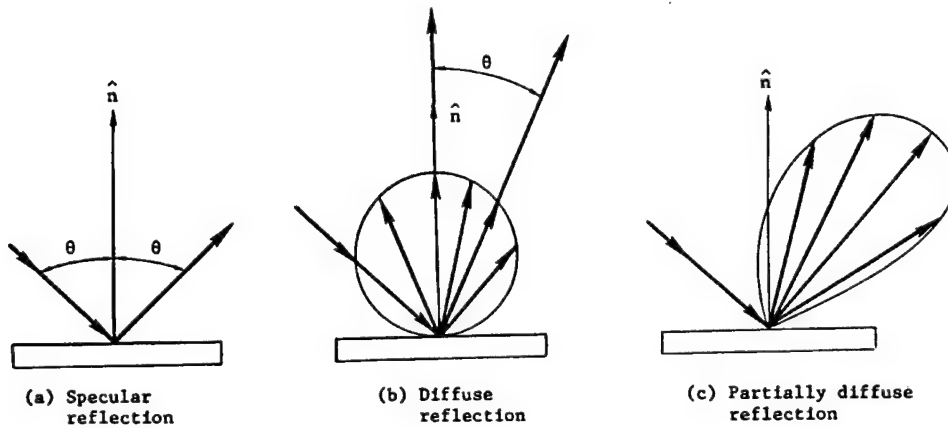


Figure 2.6    Types of Reflection [4]

In order to accurately determine the solar radiation pressure on a surface, all types of reflection must be taken into account [4]. Consider the differential area $dA$ shown in Figure 2.7. The area is oriented at an angle $\theta$ to the incoming solar radiation such that the area projected normal to the incident radiation is $dA\cos\theta$. Let $\beta$ be the coefficient of reflection (in solar spectrum) and assume the surface is opaque so that solar absorptivity is equal to $1 - \beta$. Additionally, define $\varphi$ as the portion of the reflected light which is reflected specularly, such that $\varphi = 1$ for a pure specular surface and $\varphi = 0$ for a perfectly diffuse surface. The net force imposed on the surface from solar radiation is a combination of the force due to absorption,

2-19
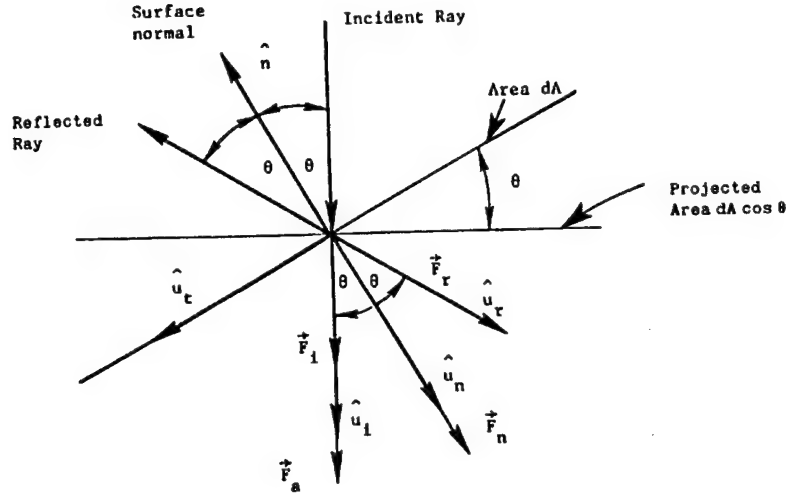
Figure 2.7    Solar Force Geometry [4]

the force due to specular reflection and the force due to diffuse reflection. The force from absorption acts in the same direction as the incident radiation and is given by

$$df_a = (1 - \beta)\frac{H}{c}dA\cos\theta\hat{u}_i \qquad (2.30)$$

The force due to specular reflection has two components; the force due to the incident radiation and the force due to reflected radiation.

$$df_{si} = \varphi\beta\frac{H}{c}dA\cos\theta\hat{u}_i \qquad (2.31)$$

$$df_{sr} = \varphi\beta\frac{H}{c}dA\cos\theta\hat{u}_r \qquad (2.32)$$

The geometry of specular reflection is such that the tangential components of $df_{si}$ and $df_{sr}$ cancel out, resulting in a net normal force.

$$df_s = (df_{si} + df_{sr})\cos\theta\hat{u}_n = 2\varphi\beta\frac{H}{c}dA\cos^2\theta\hat{u}_n \qquad (2.33)$$

2-20

The force due to diffuse reflection also has two components corresponding to the incident and reflected rays. The force due to the incident ray is given by

$$df_{di}^{\rightarrow} = (1 - \varphi)\beta \frac{H}{c} dA \cos\theta \hat{u}_i \qquad (2.34)$$

The reflected rays from a diffuse surface reflect in all directions. Applying Lambert's cosine law and integrating over the entire hemisphere through which the light is reflected reveals that the tangential components cancel out leaving a net normal force.

$$df_{dr}^{\rightarrow} = \frac{2}{3}(1 - \varphi)\beta \frac{H}{c} dA \cos\theta \hat{u}_n \qquad (2.35)$$

The total force on the surface is found by summing the components.

$$df^{\rightarrow} = df_a^{\rightarrow} + df_s^{\rightarrow} + df_{di}^{\rightarrow} + df_{dr}^{\rightarrow} \qquad (2.36)$$

Substituting (2.30), (2.33), (2.34) and (2.35) into (2.36) and combining terms gives the following expression for the total radiation force on the surface.

$$df^{\rightarrow} = \frac{H}{c} dA \cos\theta \left[ \{1 - \varphi\beta\}\hat{u}_i + \{2\varphi\beta\cos\theta + \frac{2}{3}(1 - \varphi)\beta\}\hat{u}_n \right] \qquad (2.37)$$

Torques can be generated due to solar radiation pressure acting on various parts of the satellite. As shown above, the radiation pressure causes a net force on each surface. The sum of these forces passes through the satellite center of pressure (cp). If the resultant force does not also pass through the satellite center of mass (cm), a torque is induced. The cp-cm separation represents the moment arm of this torque. The solar radiation torque is a function of this cp-cm separation, as well as the reflection characteristics of the satellite surfaces. Additionally, the satellite configuration and orbital position with respect to the sun and earth affect the radiation torque. Surfaces may be shaded by other surfaces. The whole satellite may

2-21

be shaded by the earth. These factors complicate the calculation of solar radiation torque.

In order to calculate the order of magnitude for the solar radiation torque on the IAE, the radiation pressure on the mylar canopy will be isolated. When angled towards the sun, the resultant radiation force on this surface is the dominant term in the torque equation due to its large surface area. Therefore, neglecting the pressure forces from other surfaces still provides a reasonably accurate approximation and greatly simplifies the calculations. Additionally, assume that the reflector canopy is a purely specular surface($\varphi = 1$). Equation (2.37) then reduces to

$$d\vec{f} = \frac{H}{c}dA \cos\theta[\{1 - \beta\}\hat{u}_i + 2\beta \cos\theta \hat{u}_n] \tag{2.38}$$

Recalling that the canopy is modeled as a laminar disk, integrating (2.38) over the entire area provides the resultant force through the canopy center of pressure. The normal component of this resultant force also passes through the satellite center of mass and therefore, does not create any torque. The radiation force of interest has now been reduced to

$$\vec{f}_{sri} = \frac{H}{c}A \cos\theta(1 - \beta)\hat{u}_i \tag{2.39}$$

The torque about the spacecraft center of mass is found by taking the following cross product

$$\vec{T}_{sr} = {}_{cm}\vec{r}^f X \vec{f}_{sri} \tag{2.40}$$

where ${}_{cm}r^f$ is the radius from the vehicle center of mass to the point of force application. By definition, the magnitude of the torque is the product of the two vectors' magnitude and the sine of the angle between them. Defining $\phi$ as the angle between the radius and force vectors and substituting from equation (2.39) for the magnitude of the force gives the following equation for the solar radiation torque about the IAE center of mass.

$$T_{sr} = {}_{cm}r^f[\frac{H}{c}A \cos\theta(1 - \beta)] \sin\phi \tag{2.41}$$

2-22

The radius from the IAE center of mass to the canopy center of pressure is 26.15 m along the $\hat{b}_1$ axis. This is along the same direction as the canopy surface normal vector for the given model. Therefore, $\phi$ is equal to the angle of incidence, $\theta$ as illustrated in Figure 2.8. Since the torque equation now contains the product of $\sin\theta$ and $\cos\theta$, the worst case orientation for the IAE occurs when the angle of incidence on the canopy is equal to 45 degrees. Using this orientation along with the area and material properties of the IAE model, the magnitude of the solar radiation torque is determined to be $9.01*10^{-4}Nm$. Technically, the solar radiation torque is a function of orbital altitude since the solar flux varies with distance from the sun. However, the orbital radius of most earth-orbiting satellites is so small compared to the earth's distance from the sun that the variation in solar flux is negligible, permitting the use of the solar constant in the calculations. Hence, for the purposes of this analysis, solar radiation torque is independent of orbital altitude.
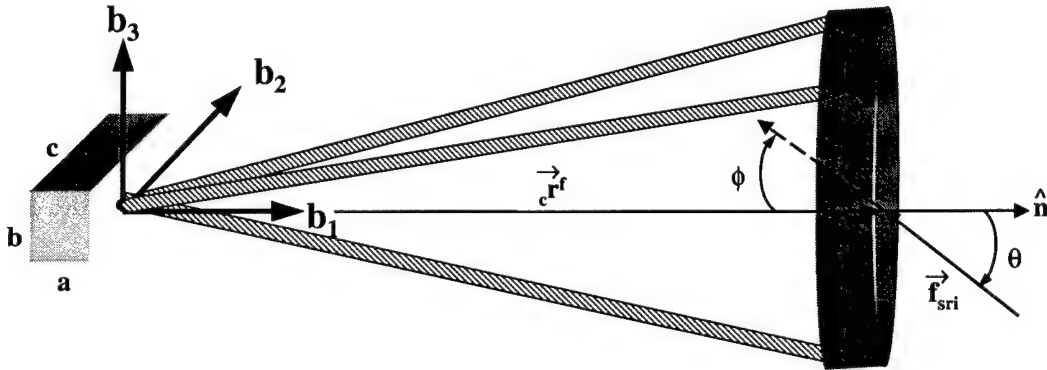


Figure 2.8    Solar Torque Calculation

## 2.6   Atmospheric Drag

Another significant disturbance on LEO satellites is aerodynamic drag. Atmospheric particles colliding with the spacecraft surfaces transfer momentum to the satellite. This momentum transfer results in a drag force which opposes the velocity

of the spacecraft. The drag force is given by

$$\vec{f_d} = -\frac{1}{2}\rho C_d A_r v \vec{v} \tag{2.42}$$

where $\rho$ is the atmospheric density, $C_d$ is the satellite coefficient of drag, $A_r$ is the presented area of the satellite, and $\vec{v}$ is the satellite velocity with respect to the atmosphere. Even at low orbital altitudes, the density of the atmosphere is small. However, the velocity of the satellite is large; thus drag force can be significant. This is especially true for satellites with large surface areas. The drag force has two main effects on a satellite's orbit [30]. First, it decreases the eccentricity of elliptical orbits. The force is greatest at perigee where atmospheric density and spacecraft velocity are at a maximum. Since the force is tangential at this point, it tends to circularize the orbit. The second effect is a diminishing semi-major axis. As the orbit becomes more circular, drag force becomes more uniform and the semi-major axis decreases more quickly. The satellite spirals in toward the earth, eventually crashing to the surface. For large satellites at low altitudes, the orbit can decay quite rapidly. Therefore, it is critical to model drag as accurately as possible in order to predict fuel consumption for orbit maintenance and the resulting satellite lifetimes. Unfortunately, modeling drag is a challenging task due to the difficulty of predicting atmospheric density and the vehicle coefficient of drag.

Atmospheric density is a function of altitude. As shown in Figure 2.9, density decreases rapidly with increasing altitude [20]. One simple approach to developing a relationship between density and altitude is to treat the atmosphere as an ideal gas in static equilibrium and sum the forces on an elementary volume of the gas. Holding temperature and gravity constant and integrating with respect to altitude provides a simple exponential model for density as a function of altitude [30].

$$\rho = \rho_o \exp\{\frac{-h}{H_o}\} \tag{2.43}$$

where

$$H_o \;=\; \frac{RT}{g}$$

$\rho_o$ is the density at the earth's surface, $h$ is the altitude, $T$ is the absolute temperature, $R$ is the gas constant and $H_o$ is called the atmospheric scale height. Scale height is a function of temperature. The scale height at the top of the earth's atmosphere is approximately 6-8km. Many atmosphere models break the atmosphere into several isothermal layers and use equation (2.43) along with the base density and local scale height to predict the density within each layer [30]. While this model will provide approximate values for density as a function of altitude, it's usefulness is limited by the simplifying assumptions of constant gravity and temperature. Gravitational acceleration decreases with altitude and the earth's atmosphere is most definitely, not isothermal.
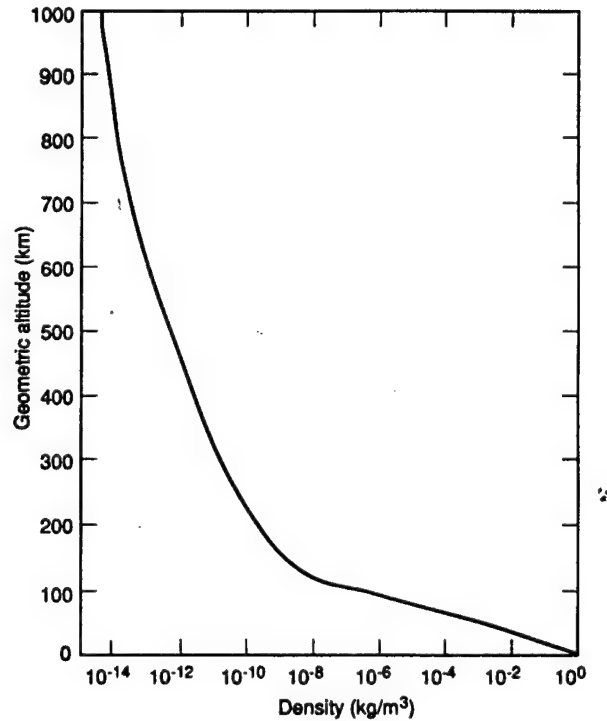


Figure 2.9    Atmospheric Density Profile [20]

The earth's atmosphere consists of several layers. The lower 100km is referred to as the homosphere and is composed collectively of the troposphere($< 10$km), the stratosphere(10km–50km), and the mesosphere(50km–90km) [26]. The temperature profile in the homosphere is characterized by a gradual temperature changes, first decreasing through the troposphere, then increasing through the stratosphere and finally decreasing again in the mesosphere. Because of the small temperature gradients, the scale height model provides reasonable values for density, especially in the stratosphere which is the most nearly isothermal region of the atmosphere [30]. However, most satellites orbit in the upper regions of the earth's atmosphere above 90 km altitude. The thermosphere(90km–500km) is characterized by rapidly increasing temperatures from its base up to the boundary with the exosphere(500km–1000km). The large temperature gradients in this region make equation (2.43) highly unreliable. Furthermore, the primary heating mechanism in this region is the absorption of solar ultra-violet(UV) and extreme ultra-violet(EUV) radiation flux which varies depending on time of day, season, and solar activity. This makes temperature difficult to predict. The average minimum temperature at the base of the thermosphere is approximately 180K, but the maximum at the exosphere boundary can vary from 500K to 1200K depending on solar activity [21].

In contrast to the other layers, the exosphere is actually isothermal. However, the very low atmospheric densities in this layer result in long mean-free paths between particles. With collision rates drastically reduced, many atmospheric particles attain sufficient kinetic energy to escape the earth's decreasing gravitational pull in this region. This is especially true of lighter particles, which have higher velocities. This phenomenon reveals another limitation of the exponential model. This shortcoming arises from the changing relative densities of the atmospheric constituents in the upper atmosphere. Air has several components, the most notable of which are molecular nitrogen and oxygen, atomic oxygen, argon and at higher altitudes, atomic hydrogen[26]. In the homosphere, atmospheric turbulence results in a

vertical mixing of the gases which causes the relative densities to remain constant. One scale height can be applied in equation (2.43) to calculate atmospheric density. However, above 90km this turbulent mixing ceases. Molecular diffusion causes the constituents to separate, with heavier elements such as molecular nitrogen, oxygen and argon occupying the lower thermosphere and lighter elements (O,He,H) becoming more prevalent in the upper thermosphere and exosphere. Hence, scale heights and densities must be calculated separately for each element.

The limitations of equation (2.43) underscore the need for a more accurate model of atmospheric density. The most widely used model is one developed circa 1970 by L.G. Jacchia. Experimental data has proven this model to predict densities in the upper atmosphere with a maximum of 10–20 percent error [20]. The major source of this error is unpredictable variation in solar intensity and geomagnetic field strength. The model assumes that the atmosphere rotates with the earth as a rigid body, and that density is a function of altitude and temperature. Temperature is given as a function of altitude and the exospheric temperature, $T_{exo}$. Furthermore, $T_{exo}$ is defined in terms of diurnal and seasonal variations in solar radiation flux, variations in the earth's magnetic field, and solar activity variations within and between solar cycles. As a result, the model requires various solar and geomagnetic indices as input as well as the position and altitude of interest. Densities are calculated separately for each atmospheric element. The overall atmospheric density is then determined by the mass weighted sum of the elemental densities. The Jacchia model will be discussed in more depth in Chapter 3.

Another difficulty in modeling drag is determining the spacecraft's coefficient of drag, $C_d$. The drag coefficient is dependent on the vehicle shape and the nature of the aerodynamic flow. A sphere in free molecular flow has a $C_d$ of approximately 2.2 [4]. The typical value used for a cylinder is 3.0. Many satellites can be approximated by one of these basic shapes. Other satellites are more complex. Observational data from the LAGEOS satellite indicates a drag coefficient of about 4.0 [15]. Predicting

the drag coefficient for more elaborate geometries is complicated by changing satellite attitude, shading effects and particle collisions with multiple surfaces. One method for analytically determining the drag coefficient is by integrating the momentum imparted to each element of the spacecraft surface by the impinging atmospheric molecules [15]. Inter-particle collisions are neglected due to the long mean free paths at orbital altitudes. This results in the following expression for drag coefficient.

$$C_d = 2 \left( 1 + \frac{1}{A_r} \int f(\theta) \cos\theta dA \right) \tag{2.44}$$

where $\theta$ is the angle between the impinging molecules and the surface normal. The presented area, $A_r$, is determined by $A\cos\theta$. The momentum transfer ration, $f(\theta)$, represents the fraction of the momentum of the reflected molecules which lies along the direction of incidence. The transfer ratio is a function of the normal($\sigma_n$) and tangential($\sigma_t$) momentum accommodation coefficients.

$$f(\theta) \quad = \quad (1 - \sigma_n)\cos^2\theta - (1 - \sigma_t)\sin^2\theta + K\sigma_n\cos\theta \tag{2.45}$$

where

$$\sigma_n \quad = \quad \frac{(p_{ni} - p_{nr})}{p_{ni}}$$
$$\sigma_t \quad = \quad \frac{(p_{ti} - p_{tr})}{p_{ti}}$$

The p's represent the components of momentum, with the subscripts i and r representing the incident and reflected components and the subscripts n and t indicating the normal and tangential components. K is a temperature constant which has been approximated at 0.038 for surfaces at 300K [15]. The accommodation coefficients are an indicator of how much of the incident momentum is converted to drag for each molecule. Zero accommodation($\sigma_n = \sigma_t = 0.0$) is equivalent to specular reflection. Particles bounce off the surface with the angle of reflection equal to the angle of

incidence. Full accommodation($\sigma_n = \sigma_t = 1.0$) is equivalent to diffuse reflection. Some of the incident particles are absorbed by the surface and later emitted in an arbitrary direction. Most surfaces have properties that fall somewhere between the two accommodation extremes. However, for the purposes of this analysis, zero accommodation will be assumed. In this case, the momentum transfer ratio simplifies to

$$f(\theta) = \cos(2\theta) \tag{2.46}$$

Substituting into equation (2.44) gives the following expression for drag coefficient.

$$C_d = 2\left(1 + \frac{1}{A\cos\theta}\int \cos(2\theta)\cos\theta dA\right) \tag{2.47}$$

In order to approximate the magnitude of drag-induced torque on the IAE, the canopy will once again be isolated. Since drag force is dependent on presented area, the force on this large surface area will dominate the torque expression when the canopy is angled in the direction of satellite velocity. Since the canopy is modeled as a flat disk, the angle of incidence of the impinging atmospheric particles with respect to the local surface normal is constant over the area of the disk. This simplifies equation (2.47) as follows.

$$\begin{aligned}
C_d &= 2\left[1 + \frac{1}{A\cos\theta}(\cos(2\theta)\cos\theta A)\right] \\
&= 2[1 + \cos(2\theta)] \tag{2.48}
\end{aligned}$$

Substituting (2.48) into (2.42) and recalling that satellite velocity in a circular orbit is equal to the square root of the gravitational parameter divided by the radius of the orbit, gives the following result for the magnitude of the drag force on the IAE canopy.

$$f_d = \rho(1 + \cos 2\theta)A\cos\theta\frac{\mu_e}{R} \tag{2.49}$$

Equation (2.40) can be used to calculate the drag induced torque, substituting the drag force in for the solar radiation force. Just like radiation pressure, the resultant drag force acts through the canopy center of pressure. Therefore, the moment arm is once again 26.15m along the $\hat{b}_1$ axis, and the angle between the moment arm and drag force vector is equal to the angle of incidence. Also, since the magnitude of the torque introduces as $\sin \theta$ term, worst case again occurs when $\theta = 45$ deg, which maximizes the product of $\sin \theta$ and $\cos \theta$. Substituting this angle into equation (2.49) along with the gravitational parameter and canopy area results in a simple expression for drag-induced torque on the IAE as a function of altitude

$$T_d = 7.961 * 10^{17} \frac{\rho}{R_e + h} Nm \tag{2.50}$$

where $\rho$ is expressed in $kg/m^3$ and $R_e$ and h are expressed in meters. Density is calculated using the scale height model. Although this model is not very accurate for the upper atmosphere, it will serve the purpose of these general order of magnitude calculations. The more accurate Jacchia method will be used for the more extensive analysis in the actual computer model. Table 2.4 lists approximate densities and the resulting drag-induced torques on the IAE at various orbital altitudes.

Table 2.4    Drag-Induced Torques

| Altitude(km) | Density($kg/m^3$) | Torque(Nm) |
|---|---|---|
| 300 | $2.4 * 10^{-11}$ | 2.86 |
| 500 | $7.0 * 10^{-13}$ | 0.081 |
| 700 | $3.6 * 10^{-14}$ | $4.05 * 10^{-3}$ |
| 1000 | $3.0 * 10^{-15}$ | $3.24 * 10^{-4}$ |
| 2000 | $1.2 * 10^{-17}$ | $1.14 * 10^{-6}$ |

## 2.7   Thermal Loads

The final environmental disturbance of concern is thermal loading. Aside from internal heat sources such as batteries and other electronic devices, satellites are subject to heat flux from environmental sources. The primary source of heat flux

on a space structure is direct solar radiation. The sun radiates energy similar to a blackbody at 6000K [20]. As previously mentioned, the average intensity of solar radiation at a distance of one AU from the sun is $1353W/m^2$. While solar radiation covers the whole electromagnetic spectrum, 97% of the energy falls in the ultra-violet, visible and near-infrared(IR) wavebands. LEO satellites are also subject to earth albedo and earth IR emissions. Earth albedo is the reflection of solar radiation from the earth's surface. The albedo ranges from 10% to 80% and is a function of local surface structure and atmospheric conditions [9]. Earth IR emissions are dependent on the temperature of the earth. An average value of $216W/m^2$ is generally assumed for intensity at the surface of the earth. Most of this energy falls in the thermal IR spectrum. All of these energy sources contribute to the thermal loading on earth-orbiting satellites. Uneven thermal loading due to shadowing effects causes temperature gradients between various satellite surfaces. In the case of an appendage which has one side illuminated and the other in shadow, cross-sectional temperature gradients arise which lead to differential thermal expansion between the two surfaces. This results in thermo-elastic deformations. Since the key design criteria for many inflatable structures will be dimensional accuracy, thermal loading could potentially degrade mission performance.

Thermally induced structural motions can be classified into one of four categories [14]. Thermal bending is a structural deformation resulting from slowly varying thermal loading. As the hot surface of an appendage expands relative to the cold surface, the appendage slowly bends away from the heat source as shown in Figure 2.10. Since the thermal load and resulting cross-sectional temperature gradient change slowly, structural accelerations and consequently torques are very small for thermal bending. When thermal loads change more rapidly, thermal snap, or thermo-elastic shock(TES), can occur. This is similar to thermal bending, but temperature gradients change rapidly. This causes rapid, non-oscillatory deformation of satellite appendages. The rapid deformation creates temporary acceleration

gradients in the appendage which induce a disturbance torque on the satellite. In some flexible appendages, rapidly changing thermal loads can induce thermal vibrations. The vibrations can involve bending or torsional motions, or a combination of both. The vibrations will generally dampen out after thermal equilibrium is reached. However, in some extreme cases, the incident heat flux couples with the structural deformations to induce unstable oscillations. This is called thermal flutter and is the worst case of thermally induced structural motion.
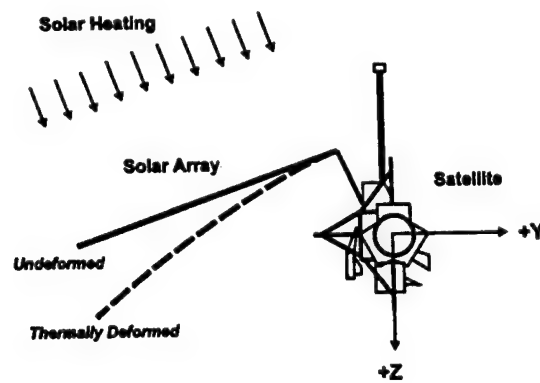


Figure 2.10    Thermal Bending in UARS [14]

In general, the thermal loading on an earth-orbiting satellite varies slowly over the course of the orbit, resulting in thermal bending of satellite appendages. However, when the satellite enters or exits eclipse, thermal loads change rapidly which can cause thermal snap or vibrations in the appendages. The acceleration gradients caused by these dynamic motions can induce a torque on the satellite resulting in attitude disturbances which may degrade performance. The Upper Atmosphere Research Satellite(UARS), shown in Figure 2.10, suffers from such disturbances [14]. Upon eclipse exit, UARS experiences acceleration transients about the roll and yaw axes of sufficient magnitude to violate the stability requirements for several of the on-board science instruments. The time history of these accelerations is consistent with the predicted response from thermal snap in the solar array. Symmetric designs such as dual wing solar arrays are often more stable, but still not impervious

to thermally-induced torques. The Hubble telescope utilizes a symmetric, dual array design. Upon eclipse exit, a 10K temperature gradient quickly arises causing structural vibrations. These vibrations last for approximately six minutes, causing maximum deflections of about ten inches[31]. These vibrations are sufficient to seriously impair Hubble's imaging capability for the duration of the disturbance. These examples highlight the need to be able to accurately predict thermally induced structural motions.

Thermal elastic shock is function of the time rate of change of the thermal gradient across a flexible structure and the mass and material properties of the appendage [31]. Figure 2.11 shows the time history of the temperature gradient across a solar panel array undergoing eclipse transition heating. The numbers were generated by a finite element analysis of a satellite in a 600km circular orbit lying in the ecliptic plane [14]. The model consisted of a rigid hub for the satellite body and a single rectangular solar array. The analysis neglected earth albedo and IR emissions, assuming that variations in direct solar radiation dominated the thermal response. At time zero, the satellite is in umbral eclipse. At ten seconds, it enters penumbra, taking 8.5 seconds to transition to full sunlight. The analysis also included equations of motion for the solar array which were integrated to determine the structural response of the solar array and satellite, shown in Figure 2.12. Note that the disturbance torque has the same profile as the second time derivative of the temperature gradient. This is the typical torque profile seen in TES, characterized by an impulsive response in one direction, followed by an exponentially decaying steplike response in the other direction [14]. For structures undergoing thermal vibrations, the same baseline pattern is seen with super-imposed oscillations which increase the amplitude and duration of the response. The peak acceleration and torque experienced are also a function of penumbral duration; the shorter the transition from umbra to full sunlight, the greater the induced torque.
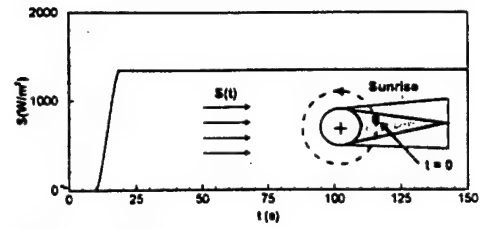
a) Solar heating history
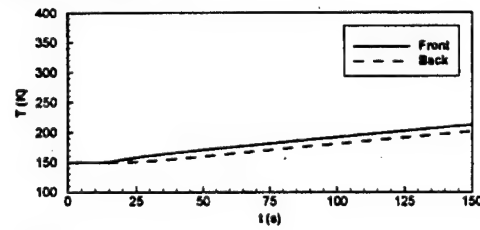
b) Surface temperatures
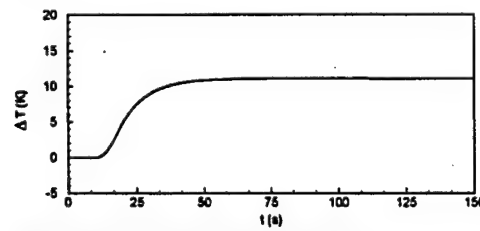
c) Through-the-thickness temperature difference
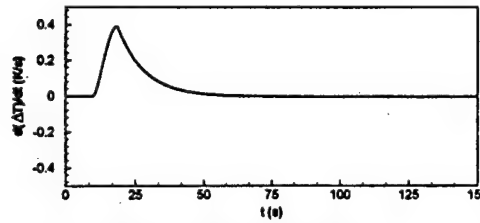
d) First time derivative of temperature difference
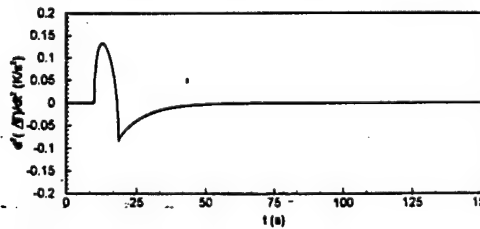
e) Second time derivative of temperature difference

Figure 2.11    Satellite Thermal Response [14]
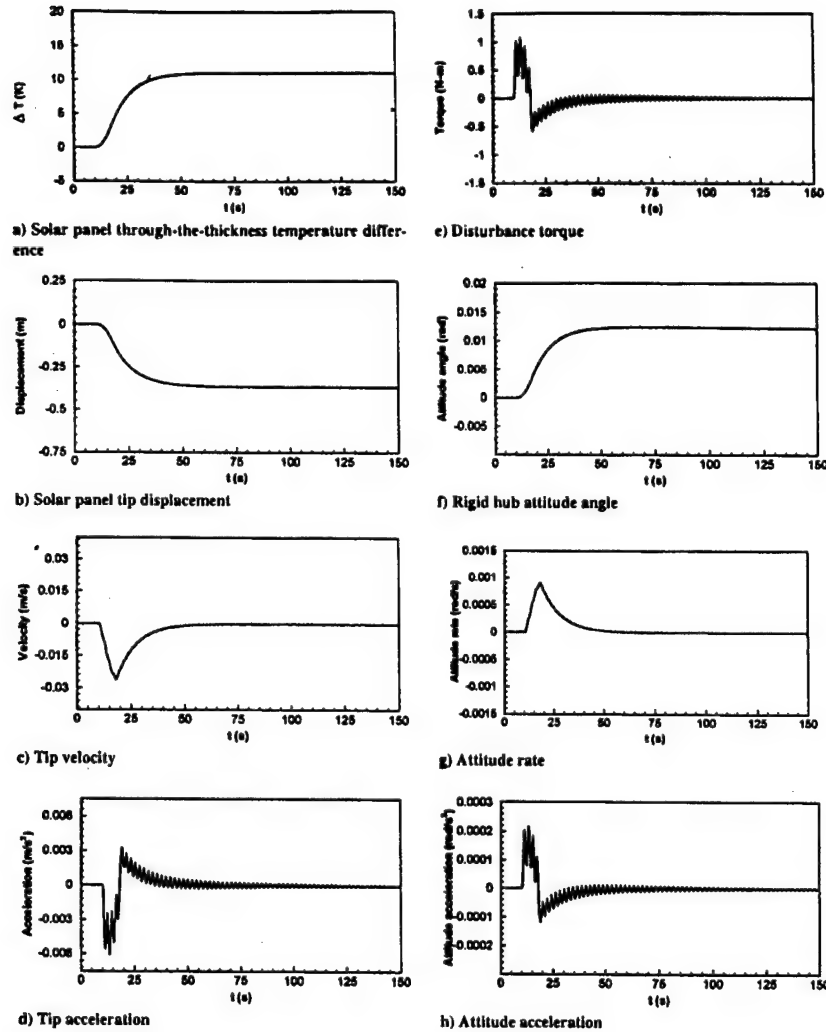
Figure 2.12    Satellite Structural Response to Thermal Snap [14]

In order to predict the thermal disturbances experienced the IAE, the torque imposed on the satellite by thermal snap of the struts during eclipse transition will be analyzed. This requires a function relating disturbance torque to the temperature gradient across a strut. Zimbelman [31] derives an equation for torque as a function of cross-sectional temperature gradient for a thin rod. This work can easily be modified for analysis of a hollow cylindrical shell such as the IAE strut. The model, shown in Figure 2.13, is based on the concept of linear thermal expansion. Given that the strut is fixed at one end (satellite body) and exposed to a radiative heat

2-35

source on one side, it will bend away from the heat source. Assuming a uniform,
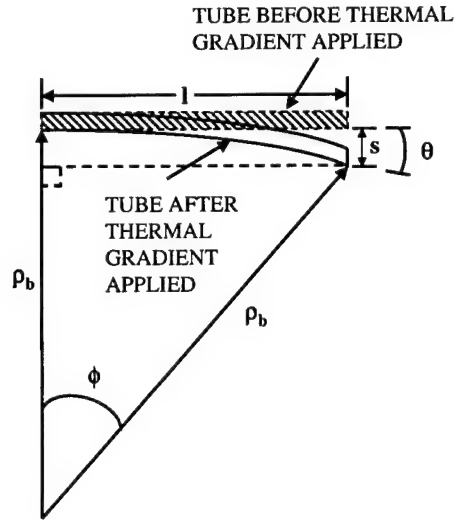


Figure 2.13    Thermal Bending Geometry [31]

linear cross-sectional temperature gradient along the length of the strut, the radius
of curvature, $\rho_b$, is given by

$$\rho_b = \frac{d}{cte * \triangle T} \tag{2.51}$$

where $d$ is the diameter of the tube, $cte$ is the coefficient of thermal expansion and
$\triangle T$ is the temperature gradient. The mass distribution of a cylindrical tube of radius
$r$, length $l$, and mass $m$ about a fixed end is given by

$$I = \frac{1}{2}mr^2 + \frac{1}{3}ml^2 \tag{2.52}$$

Since the IAE strut length(28m) is so much greater than the radius(0.18m), the strut
inertia is dominated by second term of equation (2.52). Therefore, the first term can
be neglected, simplifying the inertia calculation.

$$I = \frac{1}{3}l^2m \tag{2.53}$$

2-36

Now an inertial displacement function, $I_b$ can be defined as the strut inertia multiplied by the angular displacement of the free end relative to the fixed end as depicted in Figure 2.13.

$$I_b = \frac{1}{3}l^2 m\theta \tag{2.54}$$

Assuming that the strut deforms in a circular arc, then the arc length through which the free end travels is given by $s = l\theta$. Rearranging terms and substituting into (2.54) gives

$$I_b = \frac{1}{3}lms \tag{2.55}$$

Assuming that the angular deflection of the free end is small($\theta < 10 \deg$), then the arc length $s$ is essentially linear and can be expressed as

$$s \approx \rho_b - \rho_b \cos\phi \tag{2.56}$$

where $\phi$ is the curvature angle as shown in Figure 2.13. The curvature angle is given by

$$\tan\phi = \frac{l}{\rho_b} \tag{2.57}$$

If the length of the strut is small compared to radius of curvature($l/\rho_b < 0.176$), then $\phi$ is a small angle, simplifying equation 2.57.

$$\phi \approx \frac{l}{\rho_b} \tag{2.58}$$

Combining (2.51), (2.55), (2.56) and (2.58) yields the following expression for inertial displacement.

$$I_b = \frac{lmd}{3cte\Delta T}\left[1 - \cos\left(\frac{lcte\Delta T}{d}\right)\right] \tag{2.59}$$

Equation (2.59) describes the dynamic motion of a cylindrical tube about its fixed end in terms of its material properties and an applied temperature gradient. Taking the first time derivative of (2.59) yields the angular momentum time history of

the strut. The second time derivative gives the torque imposed on the satellite [31]. Recall though that the torque profile for thermal snap consisted of an impulsive response in one direction, followed by a steplike impulse response in the opposite direction which then exponentially decays. This step discontinuity occurs when the time rate of change of the temperature gradient hits its maximum value. This corresponds roughly to the transition of the satellite from eclipse into full sunlight [14]. Taking the the second derivative of (2.59) only captures the decaying step of the torque response, neglecting the impulsive term. However, while not completely accurate, taking this derivative and analyzing it just after eclipse exit will provide an approximate order magnitude for the maximum torque induced. So with liberal application of the chain rule and some variable manipulation, expressions for angular momentum$(H_b)$ and torque$(T_b)$ are developed.

$$H_b = \frac{-lmd\dot{\Delta T}}{3cte\Delta T^2}\left[1 - \cos\left(\frac{lcte\Delta T}{d}\right) - \frac{lcte\Delta T}{d}\sin\left(\frac{lcte\Delta T}{d}\right)\right] \quad (2.60)$$

$$T_b = \frac{\ddot{\Delta T}H_b}{\dot{\Delta T}} - \frac{2\dot{\Delta T}H_b}{\Delta T} + \frac{l^3 mcte\dot{\Delta T}^2}{3d\Delta T}\cos\left(\frac{lcte\Delta T}{d}\right) \quad (2.61)$$

In order to make use of (2.60) and (2.61), an expression for the temperature gradient is needed. Thorton [27] developed an expression for temperature distribution around a spacecraft boom. In this work, the spacecraft boom was modeled as a cylindrical shell fixed at one end. The results of this analysis can be used to determine temperature gradient as a function of time. The derivation is based on several simplifying assumptions.

- Radiation heating starts at time zero, is uniform along the length of the tube, and acquires full intensity instantaneously.

- The tube wall is so thin that the temperature gradient across it is negligible.

- Heat conduction along the tube length is negligible.

2-38

- Radiation transfer within the tube is negligible. (As mentioned in Chapter 1, radiation between opposing surfaces can be used to limit temperature gradients in inflatables. However, this assumption will be used to help generate a worst case estimate of temperature gradient.)

- Convection heat transfer inside and outside of the tube is negligible.

- Thermal properties are assumed constant.

- Incident heat flux and temperature distribution are symmetric about the tube diameter.

- Incident heat flux is independent of boom deflection.

The geometry of the analysis is shown in Figure 2.14. The temperature at any point
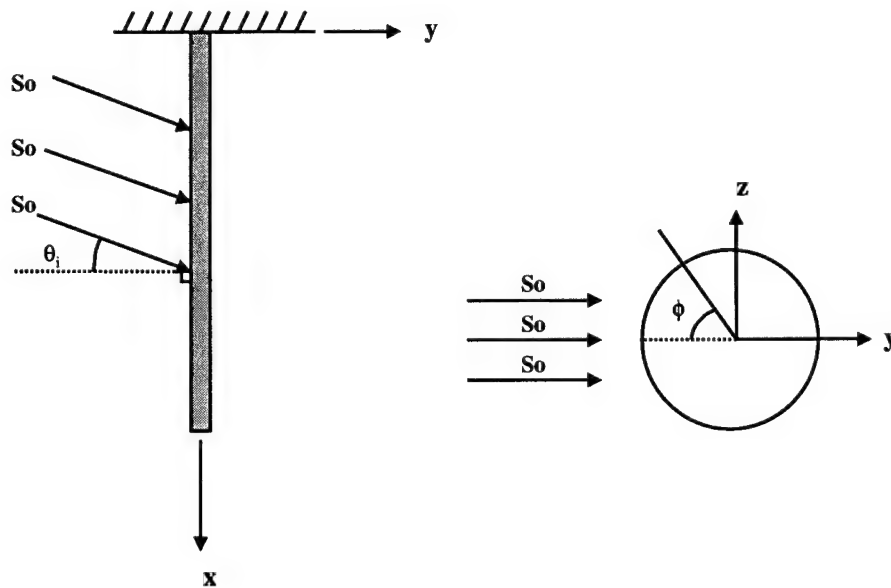


Figure 2.14    Solar Heating Geometry

on the tube wall is given by

$$T = T(\phi, t) \tag{2.62}$$

where $t$ represents time and $\phi$ is the angle between the direction of the radiation source and the radius vector to the point of interest on the tube as shown in Figure 2.14. The intensity of radiation incident at any point on the boom surface is

calculated as follows

$$S = (S_o \cos \theta_i) \delta \cos \phi \qquad (2.63)$$

where

$$\delta = 1 \quad if \quad \frac{-\pi}{2} < \phi < \frac{\pi}{2}$$
$$\delta = 0 \quad if \quad \frac{\pi}{2} \leq \phi \leq \frac{3\pi}{2}$$

The term $\delta$ is a step function which sets radiation intensity to zero on the shadowed side of the boom. Now, applying conservation of energy to a small segment of the wall and including circumferential conduction in the wall and radiation from the external surface to space results in the following [27].

$$\frac{\partial T}{\partial t} - \frac{k}{\rho c r^2} \frac{\partial^2 T}{\partial \phi^2} + \frac{\sigma \epsilon}{\rho c w} T^4 = \frac{\alpha}{\rho c w} S \delta \cos \phi \qquad (2.64)$$

where $r$ is the tube radius, $w$ is the thickness of the tube wall, $\sigma$ is the stefan boltzman constant and the other variables are the material properties defined in Table 2.1. The heat flux, $S$ is determined by $S_o \cos \theta_i$ The right hand side of equation (2.64) can be represented as a truncated fourier series.

$$S \delta \cos \phi \approx S \left( \frac{1}{\pi} + \frac{1}{2} \cos \phi \right) \qquad (2.65)$$

Now the temperature distribution can be separated into two parts. First is a uniform temperature distribution, $\bar{T}(t)$ which is independent of $\phi$. This term corresponds to a uniform heat flux, $S/\pi$, the first term of (2.65). The second part of the temperature distribution, $T_m(t) \cos \phi$, is non-uniform and corresponds to the circumferential varying term of heat flux in equation (2.65). So, the total temperature distribution is now given by

$$T(\phi, t) = \bar{T}(t) + T_m(t) \cos \phi \qquad (2.66)$$

Substituting (2.65) and (2.66) into (2.64), matching coefficients and linearizing the non-linear radiation term yields two separate differential equations for the energy balance.

$$\frac{\partial \bar{T}}{\partial t} + \frac{\sigma\epsilon}{\rho c w}\bar{T}^4 = \frac{1}{\pi}S\frac{\alpha}{\rho c w} \tag{2.67a}$$

$$\frac{\partial T_m}{\partial t} + \left(\frac{k}{\rho c r^2} + \frac{4\sigma\epsilon\bar{T}^3}{\rho c w}\right)T_m = \frac{1}{2}S\frac{\alpha}{\rho c w} \tag{2.67b}$$

It is the non-uniform temperature distribution, $T_m \cos\phi$, which creates the temperature gradient that results in torque. The uniform temperature gradient does not impact the dynamics of the boom. Therefore, for the purposes of this analysis, (2.67b) is the equation of interest. In order to simplify, the $\bar{T}$ term will be set to the steady state value which would be attained for an undeformed boom subject to heat flux S. This constant temperature is given by

$$\bar{T} = \left[\frac{1}{\pi}\frac{\alpha S}{\sigma\epsilon}\right]^{\frac{1}{4}} \tag{2.68}$$

and was found by solving (2.67a) with $\partial\bar{T}/\partial t = 0$. Substituting this value of $\bar{T}$ into (2.67b) gives

$$\frac{\partial T_m}{\partial t} + \frac{1}{\tau_c}T_m = \frac{1}{2}S\frac{\alpha}{\rho c w} \tag{2.69}$$

where

$$\frac{1}{\tau_c} = \frac{k}{\rho c r^2} + \frac{4\sigma\epsilon}{\rho c w}\left(\frac{\alpha S}{\pi\sigma\epsilon}\right)^{\frac{3}{4}}$$

$\tau_c$ is referred to as the time constant. Equation (2.69) can be rewritten as

$$\frac{\partial T_m}{\partial t} + \frac{1}{\tau_c}T_m = \frac{T^*}{\tau_c} \tag{2.70}$$

where

$$T^* = \frac{1}{2}\frac{\alpha S}{\rho c w}\tau_c$$

$T^*$ is the steady state value of $T_m$ that would be reached by an undeformed beam subject to heat flux S. It is found by solving (2.69) with $\partial T_m/\partial t = 0$. Equation (2.70) is a linear, first order differential equation which can be easily solved for $T_m$ as follows.

$$T_m(t) = \frac{T^*}{\tau_c}\int_0^t \exp\left\{\frac{q-t}{\tau_c}\right\} dq \tag{2.71}$$

Performing the integration provides an expression for the non-uniform temperature distribution as a function of time.

$$T_m(t) = \frac{\alpha S_o \cos\theta_i}{2\rho c w}\tau_c \left(1 - \exp\left\{\frac{-t}{\tau_c}\right\}\right) \tag{2.72}$$

The temperature gradient across the tube can now be found by taking the difference between the temperature on the front($\phi = 0$) and the back($\phi = \pi$).

$$\Delta T(\phi, t) = \bar{T}(t) + T_m(t)\cos(0) - [\bar{T}(t) + T_m(t)\cos(\pi)] = 2T_m(t) \tag{2.73}$$

Combining (2.72) and (2.73) provides an expression for temperature gradient as a function of time.

$$\Delta T(t) = \frac{\alpha S_o \cos\theta_i}{\rho c w}\tau_c \left(1 - \exp\left\{\frac{-t}{\tau_c}\right\}\right) \tag{2.74}$$

The first and second time derivatives of temperature gradient are easily calculated.

$$\Delta \dot{T}(t) = \frac{\alpha S_o \cos\theta_i}{\rho c w} \exp\left\{\frac{-t}{\tau_c}\right\} \tag{2.75}$$

$$\Delta \ddot{T}(t) = \frac{-\alpha S_o \cos\theta_i}{\rho c w}\frac{1}{\tau_c} \exp\left\{\frac{-t}{\tau_c}\right\} \tag{2.76}$$

Equations (2.74)–(2.76) provide the expressions for temperature gradient needed to calculate the thermal disturbance torque using equation (2.61). However, this equation for torque was developed using the assumption of a linear temperature gradient across the cylindrical shell. Therefore, in order to use (2.74)–(2.76) with (2.61), the temperature gradient given by (2.74) must be linear. This is easily verified by plotting the temperature around the circumference of the tube. Using equations (2.66), (2.68) and (2.72) along with the dimensional and material properties of the IAE strut generates the temperature profile shown in figure 2.15. The temperature gradient across the diameter of the tube is clearly linear.
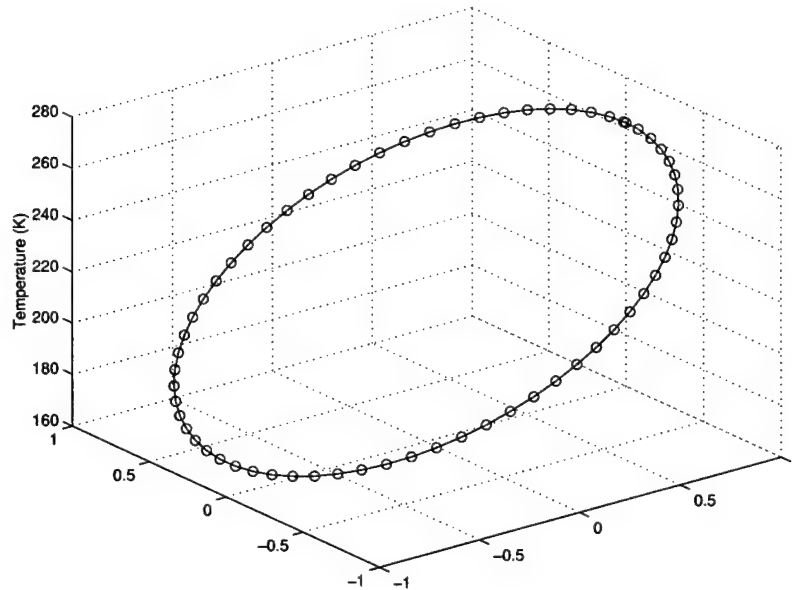


Figure 2.15    Temperature Distribution

In order to analyze the thermal response of the IAE, assume a cylindrical earth shadow with the IAE exiting eclipse at time zero. Neglect earth albedo and thermal IR emission so that $S_o$ is equal to the solar constant, $H$. Also, assume solar radiation is incident on the strut at an angle of 45 degrees as the IAE exits eclipse. Based upon earlier analysis of the thermal snap response, the maximum torque occurs when the second time derivative of $\Delta T$ is at its maximum. Visual inspection of equation (2.76) reveals that this maximum occurs at time zero. However, at time zero the

2-43

temperature gradient is zero which causes the torque equation (2.61) to be undefined due to the presence of $\Delta T$ in the denominator of several terms. This is expected since (2.61) was derived neglecting the discontinuity at time zero. To work around this discontinuity, calculate the temperature gradient and torque just after eclipse exit at t=0.01s. The disturbance torque experienced by the IAE due to radiative heating of a strut upon eclipse exit can now be calculated using equations (2.60), (2.61) and (2.74)–(2.76).

$$T_{thermal} = 5.72 * 10^{-3} Nm \tag{2.77}$$

Depending on the shadowing effects and orientation, the torque imparted to the satellite could be as high as three times this value since there are three struts. Also, note that there is no dependence on altitude. This leads to an implied assumption that no matter how high the orbit, the IAE remains in the earth shadow long enough to reach a steady state temperature distribution before exiting. In reality, in higher orbits the satellite will spend less time in umbral eclipse and more time in penumbra. Therefore, the actual thermal load changes will occur slower and be less severe. However, the simple model used above provides a reasonable approximation of a worst case scenario.

## 2.8    Summary

Now that each major environmental disturbance has been analyzed and quantified in terms of impact on the IAE, the critical loads can be determined. Figure 2.16 displays the magnitude of the torque imposed on the IAE as a function of the orbital altitude. The plot was created using the relationships developed in this chapter. It is similar to Figure 2.1, except that this plot is based upon analysis of a *large* space structure and includes thermal torques. Comparison of this plot to Figure 2.1 reveals that the torque profiles for large structures are very similar to those of small satellites. The magnitude of the torques has increased by a couple of
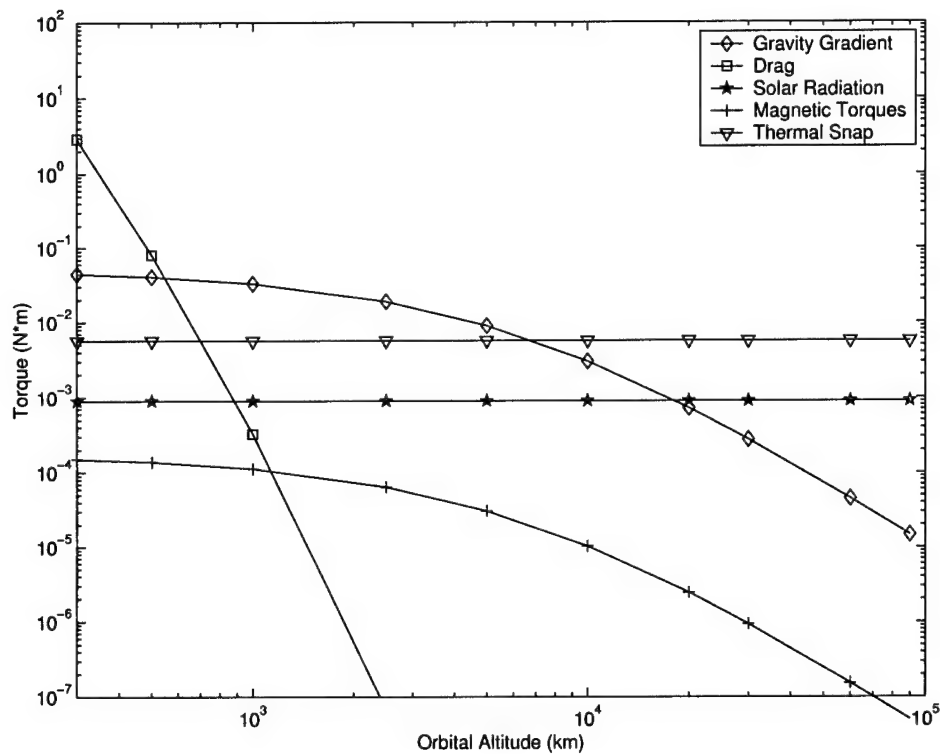
Figure 2.16    Torque vs. Altitude for the IAE

orders of magnitude, but the shape of the curves are almost identical. This makes sense. The much greater area and mass distribution of the larger structure causes the magnitude of the torques to increase. But, the relationships between altitude and the various torques remain the same. Hence, the same shape to the curves. Aside from the addition of thermal snap, the one noticeable difference between the two figures is the magnetic torque curve. While the other torques have increased by two orders of magnitude, magnetic torque has gone up by just one. The method used to approximate this torque related the magnetic moment to the area of the current loop. Therefore, while the other disturbances are directly dependent on the surface area or mass distribution of the satellite, the magnetic torque is dependent on the configuration of the electrical circuits. Depending on the configuration used, the magnetic torque curve can shift up or down relative to the other disturbances.

2-45

As expected, different disturbances dominate in different orbits. Drag is the dominant load below 700km. In fact, depending on the exact size, shape and attitude of the inflatable structure, missions in this region may be of limited use due to extremely high torques encountered and quick orbit decay times. Above 700km, gravity gradient takes over as the dominant load up to around 7000km. Above 7000km, thermally-induced torque and solar radiation pressure are the critical loads. While the predicted thermal torques are an order of magnitude higher than solar radiation torques, the thermal analysis was based on thermal snap encountered when exiting eclipse. Once thermal equilibrium has been reached after eclipse exit, or for orbits not experiencing eclipse, solar radiation pressure will be the critical load at these higher altitudes. The only load which is not ever dominant is magnetic torque. The magnetic torque follows the same curve as gravity gradient, but is two full orders of magnitude lower. Therefore, it does not constitute a critical load. Thus drag, gravity gradient, solar radiation pressure and thermal loading are the critical loads for large structures. With the critical loads identified, a disturbance model must now be developed which will calculate each load for input to a finite element analysis.

# III. Methodology

## 3.1 Overview

In Chapter 2, rough calculations on a sample satellite identified gravity gradient, drag, solar radiation pressure and thermal shock as the critical environmental loads acting on large space structures. Now that these critical loads have been determined, they must be more accurately modeled in order to determine their effects on inflatable structures. As mentioned in Chapter 1, finite element analysis provides an excellent tool for analyzing the structural response of a large body under given loading conditions. This method requires knowledge of the loads imposed on each element. Therefore, the next step in this analysis is to develop a program which allows the user to define a finite element model for a large satellite and then determines the total mechanical and thermal load on each element.

The algorithm must perform several functions. First, it must take inputs from the user to define a finite element structure. This definition will include the location and orientation of each element in terms of a body-fixed reference frame, as well as the mass, exposed surface area and material properties for each element. Next, the spacecraft's inertial position and velocity must be determined at the time of interest. Finally, the program will calculate the gravity force, drag force, solar radiation pressure and total heat flux on each element.

## 3.2 Structure Definition

Before any loads are calculated, the finite element model must be defined. This requires input from the user. The exact form of the input will be discussed in Chapter 4. However, in general, finite element codes require the user to enter node locations in terms of some known reference system and then define the elements and their nodes. Additionally, material properties must be entered for each element. The program will take these inputs from the user and then calculate other element

properties required to determine the loads imposed on the element. These properties include the element's surface area and mass, as well as the location and orientation of the element in terms of a body-fixed reference frame with its orgin at the vehicle center of mass. To reduce the complexity of the code, acceptable element definitions will be limited to point masses (1 node), line masses (2 nodes), triangular elements (3 nodes) and quadrilateral elements (4 nodes). Furthermore, the equations used to calculate area and centroid demand that at least one pair of opposing sides in the quadrilateral elements be parallel.

The surface area and mass of the elements are quickly and easily calculated. For point masses and line masses, the surface area will automatically be set to zero. The surface area for triangular and quadrilateral elements is calculated using basic geometric formulas

$$A_{triangle} = \frac{1}{2}(base * height) \tag{3.1}$$

$$A_{quad} = \frac{1}{2}(a + b) * height \tag{3.2}$$

where $a$ and $b$ are the lengths of the opposing parallel sides on the quadrilateral, and height is the perpendicular distance between them. Element mass is subsequently found by multiplying the surface area by the material density by the element thickness. For point and line masses, the total mass will be directly entered by the user.

The location of each element will be defined as the vector from the body frame orgin to the element centroid. For point masses which have only one node, the location is simply the coordinates of the node. For line masses, the centroid is the midpoint between the two nodes. Centroid calculations for triangular and quadrilateral elements are made using the geometry shown in Figure 3.1 . In a triangle, the median is defined as a line drawn from the midpoint of a side to the opposing vertex. Drawing all three medians reveals a common intersection point. This intersection point is the centroid, located two-thirds of the way along each median from the vertex to the opposing side. Calculating the centroid for a quadrilateral is done by first

drawing a diagonal between opposing nodes 1 and 3 which splits the element into two triangles, A and B. Drawing lines from the midpoint of the diagonal to nodes 4 and 2 defines medians for triangles A and B respectively. The centroid for each of these triangles is then determined as described above. The area and mass are also be calculated for each of the triangles. Treating each triangle as an equivalent point mass located at the triangle centroid, the quadrilateral centroid is located using the definition of center of mass for a system of particles, equation (2.1).
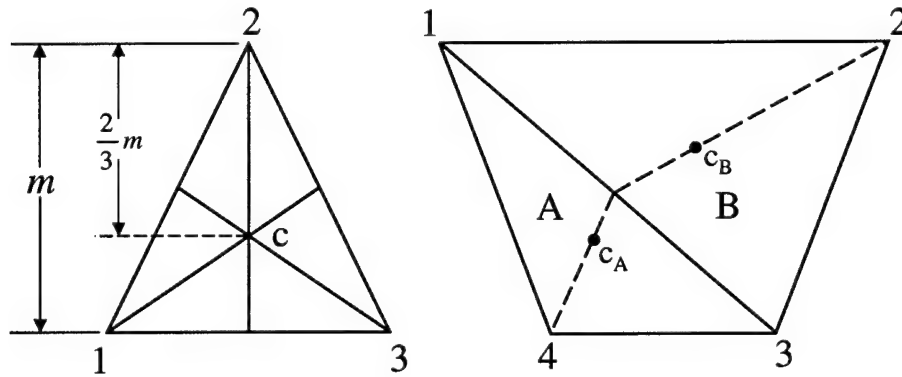


Figure 3.1    Element Centroid Determination

The orientation of an element is characterized by it's normal vector. Knowledge of the normal vector is necessary for drag, solar radiation pressure and heat flux calculations. For point and line mass elements, the normal is set equal to the zero vector. For three and four node elements, assumption of planar surfaces allows use of the cross product to determine the normal vector. Crossing any two vectors lying in the plane of the element produces a vector which is normal to the surface. However, the orientation of this normal vector may be either inward or outward depending on the order of the cross product. The orientation must be known and be consistent for all the elements on the vehicle in order to provide accurate force calculations. Therefore, a check must be performed to ensure the normal faces outward. If the vehicle is broken down by its major component, each of which is modeled by a basic

shape(i.e. rectangular boxe, cylinder, sphere etc), then a local radius vector can be defined for each element as shown in Figure 3.2(a). This vector runs from the centroid of the vehicle component on which the element resides to the centroid of the element itself. The dot product of this local radius with the element normal determines the orientation of the normal vector. A positive dot product indicates an outward normal. If the dot product is negative, the normal is oriented inward and must be reversed. However, this method will fail in the case of torus or ring type structures where the component centroid lies outside of the shape's enclosed volume. In this case, some of the outward surface normals point toward the torus centroid,
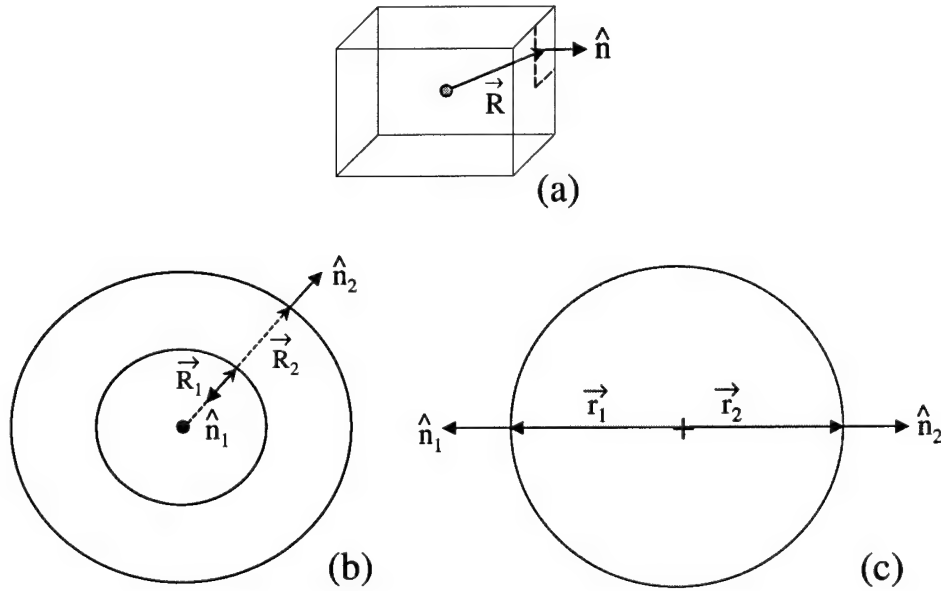


Figure 3.2    Normal Vector Orientation

while others point away as shown in Figure 3.2(b). In order to work around this difficulty, the local radius must be redefined. If a cross section of the torus is taken near the element of interest, then the local radius can be defined as the vector from the center of this cross section to the element centroid as shown in Figure 3.2(c). Now the same dot product rule can be used to ensure the normal vector is oriented outward. After ensuring element surface normals are oriented outward from the

vehicle, these vectors are normalized so that they are all have magnitude equal to one.

The calculations for element mass, surface area, location and orientation are performed for each element in the model. The output of this function is a data structure whose length is equal to the total number of input elements. Each element of the data structure contains the following nine fields: the nodes which make up the element, the location of the centroid, the outward normal vector, element area, element mass, coefficient of reflection, coefficient of specular reflection, solar absorptivity and thermal emissivity. This data is then passed back into the main program so that disturbance calculations can be made for each element.

## 3.3   Position and Velocity Calculations

In order to calculate environmental disturbances, the inertial position and velocity of the vehicle must first be determined. To simplify the process, Keplarian orbits are assumed, ignoring perturbations from secondary effects. The algorithm requires input of the six classical orbital elements: semi-major axis($a$), eccentricity($e$), inclination($i$), argument of perigee($\omega$), right ascension of the ascending node($\Omega$) and time of perigee passage($T_o$). Given these orbital elements, position and velocity can be calculated for any given time starting with Kepler's equation

$$M = \sqrt{\frac{\mu}{a^3}}\,(t - T_o) = E - e\sin E \qquad (3.3)$$

where M is the mean anomaly, t is the time of interest and E is the eccentric anomaly. For circular orbits, eccentricity is zero and the solution to Kepler's equation is simply M=E. For non-circular orbits, there is no closed form solution to Kepler's equation [30]. Therefore, the Newton-Raphson technique is used to iteratively solve for E. Convergence is achieved when the correction between successive values for E is less than $10^{-8}$. Once the eccentric anomaly is determined, the true anomaly($v$) is easily

calculated.

$$\tan\frac{\upsilon}{2} = \sqrt{\frac{1+e}{1-e}}\tan\frac{E}{2} \qquad (3.4)$$

The polar equation for a conic section is then used to determine the magnitude of the radius vector.

$$r = \frac{a(1-e^2)}{1+e\cos\upsilon} \qquad (3.5)$$

Using the true anomaly and scalar radius, a position vector can now be defined in the perifocal coordinate system as shown in Figure 3.3. This coordinate system is fixed in the orbit plane, with $\hat{p}$ lying along the eccentricity vector and $\hat{w}$ along the orbit normal. In this coordinate system, the radius vector is given by

$$\vec{r_{pqw}} = r\cos\upsilon\hat{p} + r\sin\upsilon\hat{q} \qquad (3.6)$$

Since the perifocal frame is inertial, the velocity vector is obtained by differentiating equation (3.6) with liberal use of the chain rule.

$$\vec{v_{pqw}} = \sqrt{\frac{\mu}{a(1-e^2)}}\{-\sin\upsilon\hat{p} + (e+\cos\upsilon)\hat{q}\} \qquad (3.7)$$

Although the perifocal frame is inertial, it differs between orbits. It will be more useful to have the position and velocity vectors in a standard frame such as the Earth-centered Inertial(ECI) frame. The position and velocity vectors in ECI coordinates are given by

$$\vec{r_{ijk}} = C^{pi}\vec{r_{pqw}} \qquad (3.8)$$

$$\vec{v_{ijk}} = C^{pi}\vec{v_{pqw}} \qquad (3.9)$$
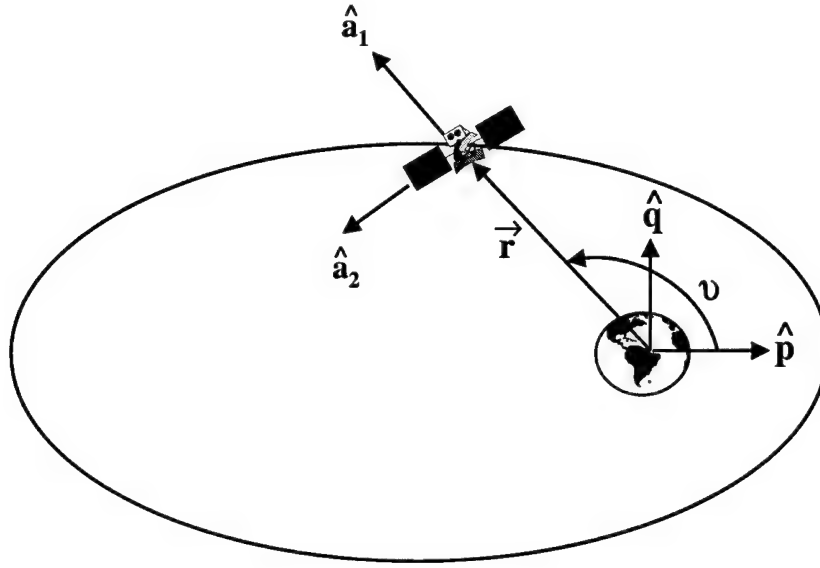
Figure 3.3    Orbit Frames

where $C^{pi}$ is a rotation matrix. Using an Euler 3–1–3 sequence, the rotation matrix is given by [30]

$$
\begin{aligned}
C^{pi} &= C_3(-\Omega)C_1(-i)C_3(-\omega) \\
&= \begin{bmatrix}
c\Omega c\omega - s\Omega cis\omega & -c\Omega s\omega - s\Omega cic\omega & s\Omega si \\
s\Omega c\omega + c\Omega cis\omega & -s\Omega s\omega + c\Omega cic\omega & -c\Omega si \\
sis\omega & cic\omega & ci
\end{bmatrix}
\end{aligned}
\tag{3.10}
$$

Position and velocity vectors are now expressed in the ECI frame. Taking these vectors to be for the satellite center of mass, the position and velocity vectors for each of the elements are easily calculated. Neglecting spinning of the spacecraft, the velocity for each of the elements is equal to the center of mass velocity. The position vector for each element is found by summing the spacecraft position vector calculated above and the radius from the vehicle center of mass to the element. This radius vector is the element location vector determined during model definition and stored in the element data strucure . Recall however, that this vector is expressed

in the body fixed frame. Prior to summing, the two vectors must be in the same reference frame. The transformation matrix from the body fixed frame to the ECI frame is determined by

$$C^{bi} = C^{pi} C^{ap} C^{ba} \qquad (3.11)$$

where $C^{ba}$ is the transformation from the body frame to the local orbit frame, $C^{ap}$ is the transformation from the local orbit frame to the perifocal frame and $C^{pi}$ is the transformation given in equation (3.10). The local orbit frame$\{\hat{a}\}$ was defined in Section 2.3 and the transformation matrix $C^{ba}$ is simply the transpose of equation (2.13). The transformation from the orbit frame to the perifocal frame is just a single axis rotation about the orbit normal given by

$$C^{ap} = \begin{bmatrix} \cos\nu & -\sin\nu & 0 \\ \sin\nu & \cos\nu & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (3.12)$$

Completing the transformation and summing the vectors yields the ECI position vector for each element. Now that the position and velocity of each element are known, the environmental disturbances can be calculated.

### 3.4  Gravitational Forces

Gravity will be the first environmental load addressed. In Chapter 2, gravity gradient calculations were made based upon a spherical earth model. In a spherical earth model, gravitational acceleration has no dependence on latitude or longitude. However, the earth is not a perfect sphere. There are asymmetries in its shape and mass distribution which cause gravity to vary with latitude and longitude as well as altitude. In order to more accurately predict the resultant forces on an orbiting satellite, the earth's gravitational potential($U_g$) will be modeled with a spherical

harmonic representation [22]

$$U_g = \frac{\mu}{r} \sum_{l=0}^{\infty} (\frac{R_e}{r})^l \sum_{m=0}^{l} P_{l,m}(\cos\phi)[C_{l,m} \cos m\lambda + S_{l,m} \sin m\lambda] \qquad (3.13)$$

where $\phi$ is the colatitude and $\lambda$ is the longitude in an earth-fixed spherical coordinate system. $C_{l,m}$ and $S_{l,m}$ are spherical harmonic coefficients of degree $l$ and order $m$, and $P_{l,m}$ are the associated Legendre functions of degree $l$ and order $m$ given by

$$P_{l,m}(x) = \frac{(1-x^2)^{\frac{m}{2}}}{2^l} \sum_{k=0}^{\frac{l-m}{2}} (-1)^k \frac{(2l-2k)!}{k!(l-k)!(l-m-2k)!} x^{(l-m-2k)} \qquad (3.14)$$

For this particular application $x = \cos\phi$.

The geopotential model shown in equation (3.13) can be broken into three constituent parts [22].

$$U_g = U_1 + U_2 + U_3 \qquad (3.15)$$

$U_1$ is the first term of the expansion where degree and order are both equal to zero ($l=m=0$). The corresponding legendre function and harmonic coefficients are equal to one. In this case, equation (3.13) simplifies to the Newtonian potential for a point or spherical mass.

$$U_1 = \frac{\mu}{r} \qquad (3.16)$$

This term is dependent only on radius. This simplification is used to derive the fundamental equation for the two body problem and was the basis for the calculations in Chapter 2. The second part of the geopotential is the set of all other terms where order is equal to zero($m=0$).

$$U_2 = \frac{\mu}{r} \sum_{l=1}^{\infty} (\frac{R_e}{r})^l P_{l,0}(\cos\phi) C_{l,0} \qquad (3.17)$$

These terms, referred to as the zonal harmonics, are dependent on both radius and latitude, but not longitude. The notation $J_l$ is often used for the zonal coefficients, where $J_l = -C_{l,0}$. The third part of the geopotential is the group of all terms where both degree and order are greater than zero.

$$U_3 = \frac{\mu}{r} \sum_{l=1}^{\infty} \left(\frac{R_e}{r}\right)^l \sum_{m=1}^{l} P_{l,m}(\cos\phi)[C_{l,m} \cos m\lambda + S_{l,m} \sin m\lambda] \qquad (3.18)$$

These terms are called the nonzonal or tesseral harmonics. In the case where degree and order are equal($l=m$), they are sometimes referred to as the sectorial harmonics. These terms are dependent on longitude. The complete geopotential can now be written as

$$
\begin{aligned}
U_g &= \frac{\mu}{r} - \frac{\mu}{r} \sum_{l=1}^{\infty} \left(\frac{R_e}{r}\right)^l P_{l,0}(\cos\phi) J_l \\
&\quad + \frac{\mu}{r} \sum_{l=1}^{\infty} \left(\frac{R_e}{r}\right)^l \sum_{m=1}^{l} P_{l,m}(\cos\phi)[C_{l,m} \cos m\lambda + S_{l,m} \sin m\lambda]
\end{aligned}
\qquad (3.19)
$$

Equation (3.19) can be used to model the gravitational field of any planet. The general form of the equation is the same for any distributed mass [20]. Developing a specific model for earth is accomplished by specifying the constants $\mu$, $R_e$, $C_{l,m}$ and $S_{l,m}$. This analysis will utilize NASA's GEM-T1 model. This model, developed by the Goddard Space Flight Center, is complete to degree and order of 36. The constants were calculated using tracking data from various satellites. Using these constants in equation (3.19), allows accurate prediction of the earth's gravitation potential at any point in a satellite's orbit. The acceleration due to gravity is then determined by taking the gradient of the potential. Working in the earth-fixed spherical coordinate system, gravitational acceleration is given by

$$\vec{a}_g = \nabla U_g = \frac{\partial U_g}{\partial r}\hat{e}_r + \frac{1}{r}\frac{\partial U_g}{\partial \phi}\hat{e}_\phi + \frac{1}{r\cos\phi}\frac{\partial U_g}{\partial \lambda}\hat{e}_\lambda \qquad (3.20)$$

where $\hat{e}_r$ points along the radius vector to the satellite, $\hat{e}_\phi$ points in the direction of increasing north latitude, and $\hat{e}_\lambda$ points in the direction of increasing east longitude. Substituting equation (3.19) into (3.20), taking the required partial derivatives and combining terms provides the following expression for gravitational acceleration [22].

$$
\begin{aligned}
\vec{a}_g = \quad & \{-\frac{\mu}{r^2} \sum_{l=0}^{\infty}(l+1)(\frac{R_e}{r})^l \sum_{m=0}^{l} P_{l,m}(\cos\phi)[C_{l,m}\cos m\lambda + S_{l,m}\sin m\lambda]\}\hat{e}_r \\
& +\{\frac{\mu}{r^2}\sum_{l=1}^{\infty}(\frac{R_e}{r})^l\sum_{m=0}^{l}\frac{\partial P_{l,m}(\cos\phi)}{\partial\phi}[C_{l,m}\cos m\lambda + S_{l,m}\sin m\lambda]\}\hat{e}_\phi \\
& +\{\frac{\mu}{r^2}\sum_{l=1}^{\infty}(\frac{R_e}{r})^l\sum_{m=1}^{l}m\frac{P_{l,m}(\cos\phi)}{\cos\phi}[-C_{l,m}\sin m\lambda + S_{l,m}\cos m\lambda]\}\hat{e}_\lambda \quad (3.21)
\end{aligned}
$$

This acceleration vector can be transformed to an earth-fixed cartesian coordinate system in the following manner.

$$
\begin{Bmatrix} a_x \\ a_y \\ a_z \end{Bmatrix} = \begin{bmatrix} \cos\phi\cos\lambda & -\sin\phi\cos\lambda & -\sin\lambda \\ \cos\phi\sin\lambda & -\sin\phi\sin\lambda & \cos\lambda \\ \sin\phi & \cos\phi & 0 \end{bmatrix} \begin{Bmatrix} a_r \\ a_\phi \\ a_\lambda \end{Bmatrix} \quad (3.22)
$$

In this rotating frame, the z axis is aligned with the earth's rotational axis and the x and y axes lie in the equatorial plane with the x axis pointed at the prime meridian. Neglecting nutation and precession of the earth's spin axis, the acceleration vector can now be easily transformed into the ECI frame.

$$
\begin{Bmatrix} a_i \\ a_j \\ a_k \end{Bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} a_x \\ a_y \\ a_z \end{Bmatrix} \quad (3.23)
$$

where $\theta$ is the Greenwich Hour Angle, the angle from the vernal equinox to the prime meridian measured in the direction of earth's rotation. (Note: The satellite radius vector must converted to spherical coordinates for input to equation (3.19). This is

accomplished by performing the above transformations in reverse order, using the transpose of the two rotation matrices.) Multiplying the acceleration by mass gives the total gravitational force on each element in the ECI frame. For planar elements, dividing by the element surface area provides an equivalent distributed gravitational load.

## 3.5  Atmospheric Drag

With the spacecraft position and velocity known, drag can also be calculated for each element. Recall from Chapter 2 that the drag force on a surface is given by

$$\vec{f_d} = -\frac{1}{2}\rho C_d A_r v \vec{v} \tag{3.24}$$

Velocity for each element has already been set equal to the velocity of the spacecraft center of mass. Therefore, determination of the drag force requires calculation of the three remaining variables in equation (3.24): the atmospheric density($\rho$), the coefficient of drag($C_d$) and the presented area of the element($A_r$).

The method used to calculate density depends on the orbital altitude. Using 2500km as a conservative estimate for the upper boundary of the earth's atmosphere, density above this altitude is set to zero. Below 90km, where turbulent mixing causes the atmospheric constituents to scale similarly, the simple exponential model for atmospheric density given in equation (2.43) is be used.

$$\rho = \rho_o \exp\{\frac{-(h - h_o)}{H_o}\} \tag{3.25}$$

Recall that this equation was developed assuming gravity and temperature constant with respect to altitude. The variance in gravity from the surface to 90km altitude is small enough to be neglected. However, the temperature changes much more rapidly with altitude and cannot be assumed constant. To work around this, the atmosphere in this region can be divided into several smaller layers. Within each layer, the

temperature change is small enough that the layer may be assumed isothermal. Such a model is described in Wertz [29]. Average values for the density at the boundaries of the layers are taken from the COSPAR International Reference Atmosphere, CIRA 72. Scale height for each layer is then found by rearranging terms in equation (3.25).

$$H_o = \frac{h_l - h_u}{\ln\left(\frac{\rho_u}{\rho_l}\right)} \tag{3.26}$$

where $h_u$, $h_l$, $\rho_u$ and $\rho_l$ are the altitudes and densities at the upper and lower boundaries of the layer. Now equation (3.25) can be used to find the density anywhere within a given layer by using the density and altitude at the base of the layer as the reference values$(\rho_o, h_o)$.

From 90km to the upper limits of the atmosphere, the Jacchia model will be used to calculate density [20]. This model is based upon the vertical diffusion equation.

$$\frac{1}{n}\frac{dn}{dz} = -\frac{mg}{kT} + \frac{1}{T}\frac{dT}{dz}(1 + \alpha) \tag{3.27}$$

where $n$ is the number density of the gas, $m$ is the molecular mass, $\alpha$ is the thermal diffusion coefficient, $g$ is the local gravitational acceleration, $k$ is the Boltzman constant and $z$ is altitude. The temperature $T$ is given by

$$T = T_{exo} - (T_{exo} - T_{zo})\exp\{-s(z - zo)\} \tag{3.28}$$

where $zo$ is a reference altitude. In this model, $zo$ is taken to be 90km. The average temperature at this reference altitude, $T_{zo}$, is equal to 183K. The reciprocal scale height of temperature, $s$, ranges from 35–70km and is determined by

$$s = 0.0291 \exp\left\{-\frac{1}{2}\left(\frac{T_{exo} - 800}{750 + 1.722 * 10^{-4}(T_{exo} - 800)^2}\right)^2\right\} \tag{3.29}$$

3-13

The exospheric temperature, $T_{exo}$, is dependent on solar activity and variations in the earth's magnetic field and can generally expressed as follows [20]

$$T_{exo} = D_v(T_{ss} + \Delta T_{ss} + \Delta T_{sa}) + \Delta T_m \qquad (3.30)$$

where

$$
\begin{aligned}
D_v &= \text{diurnal variation} \\
T_{ss} &= \text{solar cycle baseline temperature} \\
\Delta T_{ss} &= \text{variation within solar cycle} \\
\Delta T_{sa} &= \text{seasonal variation} \\
\Delta T_m &= \text{magnetic variation}
\end{aligned}
$$

Detailed expressions for these variations are found in Jacchia's 1970 paper [13]. The exospheric temperature generally ranges from 600K to around 2100K. Values less than 800K are typical for quiet solar conditions, while anything greater than 1200K is indicative of disturbed solar activity.

Determination of atmospheric density using the Jacchia model, requires integration of the vertical diffusion equation [20]. This is accomplished by first rearranging the terms of equation (3.27) as follows.

$$\frac{dn_i}{n_i} = -\left(\frac{m_i g}{kT}\right) dz - (1 + \alpha_i)\frac{dT}{T} \qquad (3.31)$$

The subscript i indicates that the equation must be solved separately for each of the four atmospheric constituents($N_2$,$O_2$,O,He), which scale separately above 90km. Also, above 500km, the density of hydrogen becomes significant and must included in the calculations. Integrating this equation will provide the number density for a specific gas at a given altitude. Integration of the first and third terms is straightforward. However, integration of the middle term is complicated by the fact that both

gravity and temperature are implicit functions of altitude. These functions must be defined and substituted into equation (3.31) before completing the integration. As shown in the previous section, the acceleration due to gravity is given by

$$g = \frac{\mu}{r^2} = \frac{\mu}{(R_e + z)^2} \tag{3.32}$$

Pulling an $R_e^2$ term out of the denominator yields

$$g = \frac{\mu}{R_e^2} \frac{1}{\left(1 + \frac{z}{R_e}\right)^2} = g_o \left(1 + \frac{z}{R_e}\right)^{-2} \tag{3.33}$$

where $g_o$ is the acceleration due to gravity at the earth's surface. Likewise, temperature is a function of altitude as given by equation (3.28). By pulling out a $T_{exo}$ term and manipulating the exponential term, the equation can be rewritten.

$$T = T_{exo}(1 - T_c e^{-sz}) \tag{3.34a}$$

where

$$T_c = \left(\frac{T_{exo} - T_{zo}}{T_{exo}}\right) e^{szo} \tag{3.34b}$$

Equations (3.34a) and (3.33) can now be substituted into (3.31) and integrated as follows

$$\int_{n_{zo}}^{n*} \frac{dn_i}{n_i} = -\frac{m_i g_o}{k T_{exo}} \int_{zo}^{z*} \frac{\left(1 - \frac{z}{R_e}\right)^{-2}}{(1 - T_c e^{-sz})} dz - (1 + \alpha_i) \int_{T_{zo}}^{T*} \frac{dT}{T} \tag{3.35}$$

where $n_{zo}$, $zo$, and $T_{zo}$ are known values at the reference altitude, $z*$ is determined from the input satellite radius vector and $T*$ is calculated using equations (3.28)-(3.30). At altitudes above 500km, temperature becomes constant($T = T_{exo}$) and the

integration is simplified.

$$\int_{n_{zo}}^{n*} \frac{dn_i}{n_i} = -\frac{m_i g_o}{kT_{exo}} \int_{zo}^{z*} \left(1 - \frac{z}{R_e}\right)^{-2} dz - (1 + \alpha_i) \ln\left(\frac{T_{exo}}{T_{zo}}\right) \tag{3.36}$$

After computing the number density for each atmospheric component, the atmospheric mass density is determined by the mass-weighted sum of the number densities.

Once atmospheric density is calculated, the coefficient of drag and the presented area for each element must be determined in order to complete the drag force computation. Recall from Chapter 2 that the coefficient of drag for a flat plate is given by

$$C_d = 2 * (1 + \cos 2\theta) \tag{3.37}$$

where $\theta$ is the angle between the element surface normal and the velocity of the element as depicted in Figure 3.4. This angle is easily calculated.

$$\theta = \cos^{-1}\left(\frac{\hat{n} \cdot \vec{V}}{|\vec{V}|}\right) \tag{3.38}$$

The presented area is also computed using $\theta$.

$$A_r = dA \cos \theta \tag{3.39}$$

When $\theta$ is greater than ninety degrees, the result of equation (3.39) is negative. This indicates that atmospheric particles are not impinging upon the element surface due to the orientation of the spacecraft. In this case, the presented surface area is set to zero. The total drag force can now be calculated using equation (3.24). The distributed drag load is obtained by dividing out the presented area.
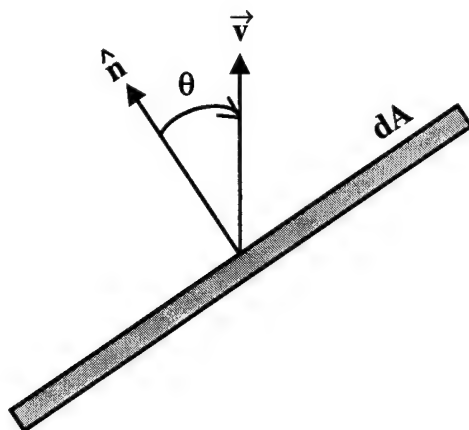
Figure 3.4    Drag Calculations

## 3.6  Solar Intensity Calculations

Before solar radiation pressure and heat flux calculations can be made, the sun-earth-satellite geometry must be analyzed to determine if the satellite is in eclipse. The dual-conic eclipse model, illustrated in Figure 3.5, is used. A solar intensity coefficient($S_i$) must be calculated to indicate the portion of the solar disk visible to the satellite. This unitless coefficient ranges from zero to one, with a value of zero corresponding to umbral eclipse and a value of one signifying no eclipse. Any mid-range value indicates that the vehicle is in penumbral eclipse. Once the solar intensity coefficient is calculated, it is multiplied by the solar constant to determine the actual intensity of solar radiation incident on the satellite surfaces.

The first step in describing the sun-earth-satellite geometry is to determine the position of the sun relative to the earth. Since the ECI frame translates with the earth, an observer at the orgin of this frame would perceive the sun as orbiting around the earth. The elements of this apparent orbit are listed in Table 3.1. The asterisked elements are given for J2000 which is shorthand notation for 12:00:00 Universal Time on January 1, 2000. The relative position of the sun and earth at this time can be found in a variety of references. JPL publishes the ephemeris for all of the celestial bodies in the solar system at http://ssd.jpl.nasa.gov. The output
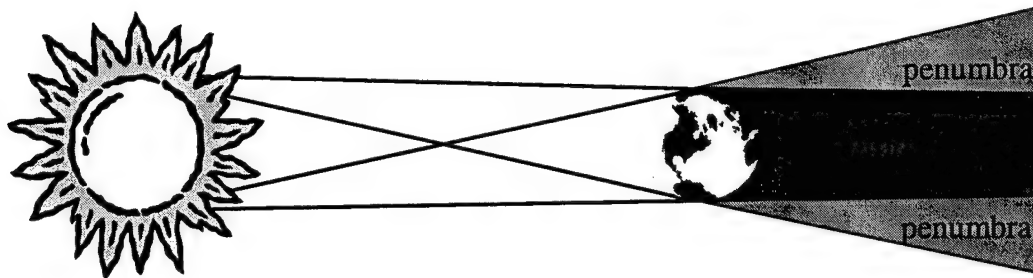
Figure 3.5     Dual-conic Eclipse Model

is available in several different formats and coordinate systems. Using the sun's position relative to the earth at the J2000 reference time and the orbital parameters in Table 3.1, the sun's position in the ECI frame can be predicted for any given time. This requires calculating the obliquity of the ecliptic and the true anomaly at the time of interest. Meeus [18] provides the following algorithm for these calculations which is accurate to within 0.01 deg.

Table 3.1     Sun-Earth Orbital Parameters [18]

| Sun-Earth Orbital Parameters | |
|---|---|
| Semimajor Axis($a$) | 1.000 AU |
| Eccentricity($e_o$) | 0.016708634* |
| Obliquity of Ecliptic($\varepsilon_o$) | 23.43929 deg * |
| Right Ascension of the Ascending Node($\Omega$) | 0.0 deg |
| Argument of Perigee($\omega$) | 282.940308 deg |
| Mean Anomaly($M_o$) | 357.52911 deg * |

Obliquity of the ecliptic, $\varepsilon$, is the angle between the ecliptic plane and the earth's equatorial plane. It is analogous to the inclination of the sun's apparent orbit around the earth. Due to the precession of the earth's spin axis, the obliquity of the ecliptic varies with time as follows.

$$\varepsilon = \varepsilon_o - 0^o.013004T + 1^o.64 * 10^{-7}T^2 + 5^o.04 * 10^{-7}T^3 \qquad (3.40)$$

3-18

The elapsed time since J2000, $T$, is expressed in julian centuries and is calculated by

$$T = \frac{JD - 2451545.0}{36525} \qquad (3.41)$$

where $JD$ is the Julian date at the time of interest. Substituting equation (3.41) into (3.40) gives the new obliquity of the ecliptic.

Calculating the true anomaly is slightly more involved. The first step is to compute the mean anomaly as follows.

$$M = M_o + 35,999°.05029T - 0°.0001537T^2 \qquad (3.42)$$

Normally, Kepler's equation is solved using the Newton-Raphson method to determine true anomaly. However, for orbits with small eccentricity, the true anomaly can be found using the equation of center, C. The equation of center represents the difference between true anomaly and mean anomaly, and is calculated directly from eccentricity and mean anomaly.

$$C = \left( 2e - \frac{e^3}{4} \right) \sin M + \frac{5}{4}e^2 \sin 2M + \frac{13}{12}e^3 \sin 3M \qquad (3.43)$$

Eccentricity of the sun's orbit as viewed from Earth varies with time and is given by

$$e = e_o - 0.000042037T - 0.0000001267T^2 \qquad (3.44)$$

Substituting equation (3.44) into (3.43) gives the equation of center in radians. Converting to this value to degrees and adding it to the mean anomaly gives the true anomaly.

The ECI radius from the earth to the sun $(\vec{r_{es}})$ can now be calculated using the methods outlined in Section 3.3, equations (3.5), (3.6), (3.8) and (3.10). Since $\hat{i}$ is defined by the point where the sun crosses the equator going south to north, right

3-19

ascension of the ascending node for the sun's apparent orbit is always zero. This simplifies equation (3.10) to

$$C^{pi} = \begin{bmatrix} \cos\omega & -\sin\omega & 0 \\ \cos\varepsilon\sin\omega & \cos\varepsilon\cos\omega & -\sin\varepsilon \\ \sin\varepsilon\sin\omega & \sin\varepsilon\cos\omega & \cos\varepsilon \end{bmatrix} \qquad (3.45)$$

where obliquity of the ecliptic has been substituted in for inclination. Once $\vec{r_{es}}$ is determined, subtracting the satellite's position vector $(\vec{r_{sat}})$ gives the vector from the satellite to the sun $(\vec{r_{svs}})$. The sun-earth-satellite geometry, shown in Figure 3.6, is now sufficiently specified to make solar intensity calculations.
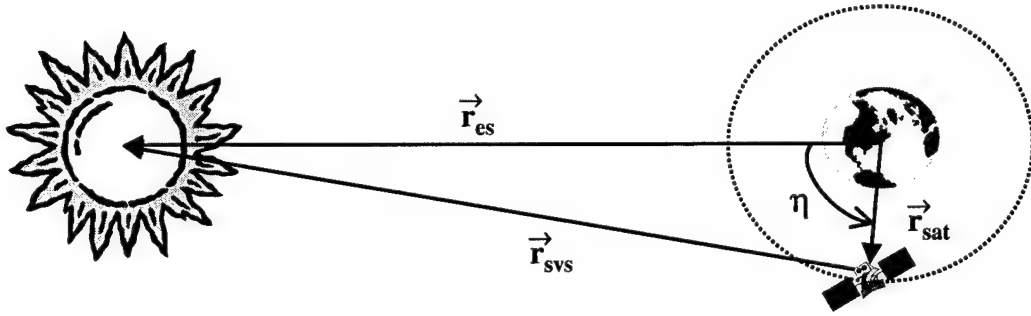


Figure 3.6    Sun-Earth-Satellite Geometry

The intensity of the solar radiation incident on a satellite is proportional to the area of the solar disk that is visible to the satellite [24]. Mathematically, this is determined by

$$S_i = 1 - \frac{A_b}{A_s} \qquad (3.46)$$

where $A_s$ is the total surface area of the solar disk and $A_b$ is the solar disk area which is blocked from the satellite's view by the earth. The portion of the solar disk blocked from view is determined by the angular radii of the earth and sun, and the angular separation between the two bodies as viewed from the satellite. These

measurements are illustrated in Figure 3.7. The angular radii of the sun and earth
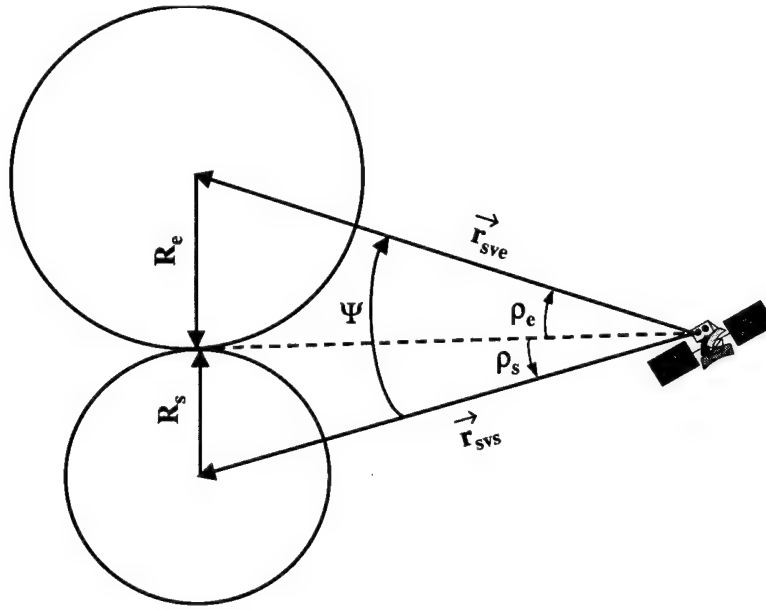


Figure 3.7    Eclipse Prediction

are given by

$$\rho_s = \sin^{-1}\left(\frac{R_s}{|\vec{r_{svs}}|}\right) \tag{3.47}$$

$$\rho_e = \sin^{-1}\left(\frac{R_e}{|\vec{r_{sve}}|}\right) \tag{3.48}$$

where the satellite-earth vector, $\vec{r_{sve}}$, is just the opposite of $\vec{r_{sat}}$. For earth orbiting satellites, $\rho_e > \rho_s$ as long as the radius of the satellite's orbit is less than $1.4 * 10^6$ km. This means that the earth appears larger to the satellite and is capable of totally obscuring the sun, causing umbral eclipse. Given the angular radius, the total solar disk area can now be calculated.

$$A_s = \pi \rho_s^2 \tag{3.49}$$

The solar disk area blocked from the satellite's view by the earth is dependent on $\Psi$, the angular separation between the earth and sun.

$$\psi = \cos^{-1}\left(\frac{\vec{r_{svs}} \cdot \vec{r_{sve}}}{|\vec{r_{svs}}||\vec{r_{sve}}|}\right) \tag{3.50}$$

3-21

When $\psi \geq (\rho_s + \rho_e)$ the earth is not obscuring the solar disk at all. $A_b$ is equal to zero, resulting in a solar intensity coefficient of one. Conversely, when $\psi \leq (\rho_e - \rho_s)$, the sun is totally obscured by the earth. In this case, $A_b = A_s$ and the solar intensity coefficient is equal to zero. When $\psi$ falls between these boundaries, the earth is partially obscuring the solar disk. In order to determine the portion of the disk blocked from the satellite's view, two intermediate variables, $x$ and $h$, are introduced [24].

$$x = \frac{1}{2}(\rho_s + \rho_e + \psi) \tag{3.51}$$

$$h = \frac{2}{\psi}\sqrt{x(x - \psi)(x - \rho_s)(x - \rho_e)} \tag{3.52}$$

If $\psi^2 \geq (\rho_e^2 - \rho_s^2)$, then $A_b$ is given by

$$A_b = \left\{\sin^{-1}\left(\frac{h}{\rho_s}\right)\right\}\rho_s^2 + \left\{\sin^{-1}\left(\frac{h}{\rho_e}\right)\right\}\rho_e^2 - h\psi \tag{3.53}$$

If $\psi^2 < (\rho_e^2 - \rho_s^2)$, then $A_b$ becomes

$$A_b = \rho_e^2\sin^{-1}\left(\frac{h}{\rho_e}\right) + \left\{\pi - \sin^{-1}\left(\frac{h}{\rho_s}\right)\right\}\rho_s^2 - h\psi \tag{3.54}$$

These expressions for $A_b$ return a value between zero and $A_s$, resulting in a solar intensity between zero and one [24]. This corresponds to penumbral eclipse. The preceding equations remain valid as long as the spacecraft orbit radius is less than the threshold value specified above $(1.4 * 10^6 km)$.

## 3.7 Solar Radiation Pressure

Now that the solar intensity coefficient is known, it can be inserted into equation (2.37) to obtain the force due to solar radiation pressure on each element.

$$d\vec{f_{srp}} = S_i\left(\frac{H}{c}\right)dA\cos\theta\left[\{1 - \varphi\beta\}\hat{u}_i + \{2\varphi\beta\cos\theta + \frac{2}{3}(1 - \varphi)\beta\}\hat{u}_n\right] \tag{3.55}$$

where $\theta$, the angle of incidence of the incoming radiation is determined by

$$\theta = \cos^{-1}\left(\frac{\hat{n} \cdot \vec{r}_{svs}}{|\vec{r}_{svs}|}\right) \tag{3.56}$$

The geometry of the interaction is illustrated in Figure 2.7. The vector $\hat{u}_n$ is simply the opposite of the element surface normal, $\hat{n}$. Likewise, $\hat{u}_i$ is a unit vector in the opposite direction of $\vec{r}_{svs}$. Making the appropriate substitutions, the force due to solar radiation can now be expressed as

$$d\vec{f}_{srp} = -S_i\left(\frac{H}{c}\right)dA\cos\theta\left[\{1 - \varphi\beta\}\frac{\vec{r}_{svs}}{|\vec{r}_{svs}|} + \{2\varphi\beta\cos\theta + \frac{2}{3}(1-\varphi)\beta\}\hat{n}\right] \tag{3.57}$$

The solar radiation pressure, or the distributed load is obtained by dividing out the $dA\cos\theta$ term. Of course, due to the orientation of the vehicle, some elements may be facing away from the sun. This is easily determined by examining $\theta$. If $\theta \geq 90\,\text{deg}$, then the element is facing away and solar radiation pressure is set to zero.

Recall from Chapter 2 that $H$ represents the solar constant, given to be $1353 W/m^2$. The is the average solar flux at a distance of one AU from the sun. In reality, the solar flux at the upper edge of the earth's atmosphere varies with the level of solar activity and the distance between the sun and earth. While solar activity is difficult to predict, the changing distance between the earth and sun is easily modeled. Treating the sun as an isotropic source with an average output power of $3.8 * 10^{26}$ watts, the solar flux at the edge of earth's atmosphere is given by

$$H_s = \frac{3.8 * 10^{26}}{4\pi R^2}\frac{W}{m^2} \tag{3.58}$$

where $R$ is the distance (in meters) from the earth to the sun [2]. This value for solar flux accounts for the changing distance between the two bodies as the earth travels through its slightly elliptical orbit. The subscript $s$ has been added to the solar flux

$(H_s)$ to distinguish it from the solar constant. Using $H_s$ in equation (3.57) instead of the solar constant will provide a more accurate model.

## 3.8   Heat Flux

The final disturbance to be calculated is the heat flux into each element from environmental sources. There are three main sources of radiance on earth-orbiting satellites: direct solar radiation, earth-reflected solar radiation and thermal radiation emitted from the earth [20]. While direct solar radiation is the dominant source, reflected solar radiation and earth-emitted thermal energy can be significant contributors for LEO satellites. In the case of satellites in umbral eclipse, earth emissions are the only source of external heat flux into the spacecraft. Therefore, heat flux from each source must be considered.

The total energy from direct solar radiation incident upon a spacecraft element is determined by

$$G_{ds} = S_i H_s dA_p \qquad (3.59)$$

where $H_s$ is defined in the previous section and $G$ is called the irradiance. The element area projected in the direction of the satellite-sun vector, $dA_p$, is equal to the element area$(dA)$ times the cosine of the incidence angle$(\theta)$, determined by equation (3.56). As was done in the solar radiation pressure calculations, the orientation of the element is checked to determine if it is facing away from the sun. If $\theta \geq 90\,\mathrm{deg}$, then $G_{ds}$ is set to zero.

The irradiance from earth-emitted thermal energy is obtained by

$$G_{et} = F_{et} H_e dA \qquad (3.60)$$

where $H_e$ is the earth constant. It represents the average intensity of the emitted thermal radiation at the earth's surface. Treating the earth as a greybody which is

re-radiating absorbed solar energy, the earth constant is approximated by

$$H_e = H_s \frac{1 - a}{4} \tag{3.61}$$

where $a$ is the earth's albedo [20]. As mentioned in Chapter 2, albedo can range from 0.1–0.8, depending on surface structure and atmospheric conditions. In order to keep the model simple, the average earth albedo of 0.36 is used [9].

The configuration or view factor, $F_{et}$ accounts for the fraction of the total energy leaving the earth which is incident on the spacecraft element. This view factor considers not only the orientation of the element, but also the distance from the earth to the satellite. Since the elements in this model are planar, $F_{et}$ can be approximated by the standard view factor between a finite plate and a sphere [12]. The geometry used to calculate this view factor is illustrated in Figure 3.8. The
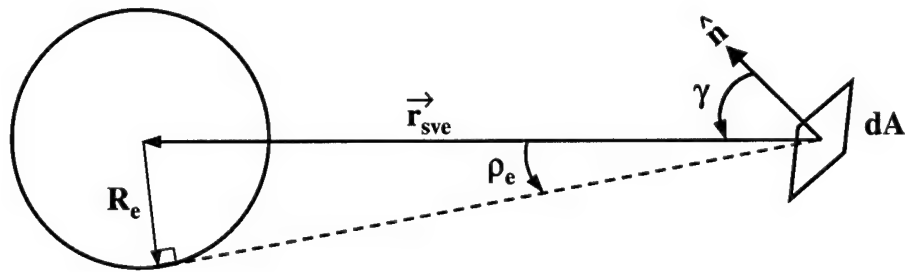


Figure 3.8    View Factor Geometry

angular radius of the earth($\rho_e$) was defined in equation (3.48). The angle between the element normal and the satellite-earth vector, $\gamma$, is easily computed by

$$\gamma = \cos^{-1}\left(\frac{\vec{r_{sve}} \cdot \hat{n}}{|\vec{r_{sve}}|}\right) \tag{3.62}$$

Now define variable $X$ as the ratio of the satellite radius to the earth radius.

$$X = \frac{|\vec{r_{sve}}|}{R_e} \tag{3.63}$$

If $\frac{\pi}{2} - \rho_e \leq \gamma \leq \frac{\pi}{2} + \rho_e$, then the view factor is given by

$$F_{et} = \frac{1}{2} - \frac{1}{\pi}\sin^{-1}\left[\frac{(X^2-1)^{\frac{1}{2}}}{X\sin\gamma}\right]$$
$$+ \frac{1}{\pi X^2}\{\cos\gamma\cos^{-1}[-(X^2-1)^{\frac{1}{2}}\cot\gamma] - (X^2-1)^{\frac{1}{2}}[1 - X^2\cos^2\gamma]^{\frac{1}{2}}\} \tag{3.64}$$

If $\gamma < \frac{\pi}{2} - \rho_e$, then the view factor simplifies to

$$F_{et} = \frac{\cos\gamma}{X^2} \tag{3.65}$$

Finally, if $\gamma > \frac{\pi}{2} + \rho_e$, this indicates that the element is facing away from the earth. Consequently, the view factor is equal to zero. Using the appropriate view factor, the irradiance from earth-emitted thermal energy is computed.

The irradiance from earth-reflected solar radiation is determined by

$$G_{er} = aH_sF_{er}dA \tag{3.66}$$

where the earth's albedo ($a$) is again set equal to 0.36. The view factor from the spacecraft element to the sunlit portion of the earth, $F_{er}$, is approximated by [20]

$$F_{er} = F_{et}\cos\eta \tag{3.67}$$

where $\eta$ is the angle between the earth-sun vector and the satellite position vectors as illustrated in Figure 3.6. This angle is found through the dot product of the two

vectors.

$$\eta = \cos^{-1}\left(\frac{\vec{r_{es}} \cdot \vec{r_{sat}}}{|\vec{r_{es}}||\vec{r_{sat}}|}\right) \tag{3.68}$$

If $\eta > 90\deg$, then $F_{er} = 0$.

Heat flux from each source is now calculated by multiplying the irradiance by the appropriate coefficient of absorption. The coefficient is dependent on the material composing the element and the wavelength of the incident radiation. For direct and reflected solar radiation, the material's average coefficient of absorption across the UV and visible wavelengths is used. For earth-emitted thermal radiation, the element's absorption coefficient in the thermal IR band must be used. Once the heat flux from each source is determined, they are summed to find the total heat flux into the element. It is important to note that this is the total heat flux into the element from environmental sources. It is not the net heat flux into or out of the element. In order to determine the net heat flux, radiation from the element to space must be included. This radiation heat transfer is dependent on the temperature of the element. The element temperature in turn, is dependent not only on the incident radiation from the environment, but also on conduction from surrounding elements, radiation from opposing surfaces and heat sources within the spacecraft. The computation of the element temperature and corresponding radiation is beyond the scope of this analysis.

## 3.9  Summary

The critical environmental loads have now all been calculated. These calculations are placed within two loops in the code. The outer loop is a time loop, calculating positions and loads at each time of interest. The inner loop is an element loop which calculates the loads for each element at each time step. The finite element definition is performed outside both loops. Position and velocity calculations for the satellite center of mass and the atmospheric density computation code are placed just inside the outer loop. Therefore, these parameters are calculated just

once per time step. All other calculations are located inside the inner loop, obtaining unique values for each element. The final code, along with sample output and user instructions are presented in the next chapter.

# IV. Code Description and Analysis

## 4.1 Overview of Code

Using the methodology outlined in Chapter 3, a disturbance model was built in MATLAB v5.3.1. The complete code is included in Appendix A. The routines in Table 4.1 were developed specifically for this disturbance model.

Table 4.1    Unique MATLAB Routines

| Title | Description |
|-------|-------------|
| **forces.m** | main executable program which propagates the model through time, calling the appropriate subroutines to calculate the loads on each element |
| **newmodel.m** | template for user to enter orbital, material and structural properties needed to create a model and run an analysis |
| **structure.m** | takes the node and element definitions input by the user and builds the finite element model |
| **randv.m** | calculates the satellite's instantaneous position and velocity vectors given the classical orbital elements and current time |
| **sunvec.m** | defines the sun's location in ECI coordinates |
| **density.m** | gathers/formats data needed to determine atmospheric density |
| **grav.m** | determines the gravitational force on each element |
| **solintens.m** | calculates the solar intensity coefficient |
| **solpress.m** | determines the solar radiation pressure on each element |
| **drag.m** | calculates the aerodynamic drag on each element |
| **thermal.m** | computes heat flux into each element from the environment |
| **graphic2.m** | creates graphic representation of the loads on each element |

In addition to the unique files listed above, the algorithm uses several routines from the *Spacecraft Control Toolbox v3.0*, developed by Princeton Satellite Systems. These routines are described in Table 4.2.

Table 4.2    Spacecraft Control Toolbox Routines

| Title | Description |
|---|---|
| Date2JD.m | converts a universal date and time to julian date format |
| GMSTime.m | calculates Greenwich Mean Sidereal Time, the angle between the prime meridian and vernal equinox measured in the direction of earth's rotation |
| JD2Date.m | converts a julian date to universal date/time format |
| R2LatLon.m | calculates the latitude and longitude of the satellite subpoint given the position vector in an earth-fixed, cartesian coordinate system |
| AtmDens2.m | determines atmospheric density using the scale height model |
| AtmJ70.m | calculates atmospheric density using the Jacchia 1970 model |
| Agravity.m | computes gravitational acceleration in spherical coordinates using NASA's GEM-T1 model |
| JSp2Cart.m | Calculates the Jacobian for converting from spherical to cartesian coordinates |

At each timestep, the model calculates the total force from gravity, drag and solar radiation pressure on each element, as well as a distributed load from each disturbance. These loads are stored in the matrices defined in Table 4.3. Point and line masses were assumed to have zero area, resulting in zero drag and solar radiation pressure. Therefore, the total distributed load (dP) on these elements is equal to the distributed gravity load with corresponding units. Each of the matrices in Table 4.3 is three dimensional $(3 \times j \times i)$ with $j$ equal to the number of elements and $i$ corresponding to the number of time steps. All loads are in terms of the body-

Table 4.3    Mechanical Load Matrices

| Variable | Load Description (units) |
|----------|--------------------------|
| **dfg** | total force due to gravity $(N)$ |
| **dfd** | total force due to drag $(N)$ |
| **dfsr** | total force due to solar radiation $(N)$ |
| **df** | sum total force on each element from all disturbances $(N)$ |
| **dPg** | the distributed gravity load. For planar elements, this is equal to the gravity force divided by the element surface area $(N/m^2)$. For line elements, it is the force per unit length $(N/m)$, and for point masses it is equal to total force $(N)$. |
| **dPd** | the distributed drag load $(N/m^2)$ |
| **dPsr** | the total solar radiation pressure $(N/m^2)$ |
| **dP** | sum total distributed load on each element $(N/m^2)$ |

fixed reference frame. The program also computes and stores the heat flux into each element from environmental sources. These thermal loads are stored in the matrices described by Table 4.4. These matrices are two dimensional $(j \times i)$ with $j$ and $i$ as defined above.

Several other variables of importance are also generated by the simulation. *Time* is a row vector containing the julian date/time at each time step. *Node* is a data structure containing the body-fixed coordinates of each node in the model. *Element* is another data structure which includes each element's location, orientation and material properties. All of these variables along with the load matrices described above are written to the *forces.mat* file, which is automatically saved in the MATLAB work directory.

Table 4.4    Thermal Load Matrices

| Variable | Load Description (units) |
| --- | --- |
| dQs | heat flux from direct solar radiation (W) |
| dQer | heat flux from earth-reflected solar radiation (W) |
| dQet | heat flux from earth-emitted thermal radiation (W) |
| dQe | sum total heat flux from environmental sources (W) |

## 4.2  User Input

All required inputs are consolidated into a single file, *Newmodel.m.* Each new simulation requires the specification of the following data: orbital parameters, environmental indices, simulation run parameters, material properties, structure component descriptions, node locations, and element properties. A sample input file is shown in Appendix B.1.

The orbit is defined by the six classical orbital elements, $a$, $e$, $i$, $\Omega$, $\omega$, and $T_o$. All elements must be specified. Therefore, for circular or equatorial orbits, small values for inclination and eccentricity (0.000001) must be used along with arbitrary values for right ascension of the ascending node and argument of perigee. All angles are entered in degrees. The time of perigee passage must be in universal time. Semi-major axis is input in kilometers.

To determine the atmospheric density using the Jacchia model, the algorithm requires four solar/geomagnetic indices. The planetary geomagnetic index, $A_p$, is a planetary average of variations in the earth's magnetic field. The daily 10.7cm solar flux, $f_{10.7}$, is a measure of solar activity. The 81-day mean of $f_{10.7}$ ($f_{hat}$) and the value of $f_{hat}$ 400 days prior to the calculation ($f_{400}$) must also be specified. Historical records and current values for these geomagnetic and solar activity indices are published by the National Oceanic and Atmospheric Administration at *http://www.sec.noaa.gov/data/geomag.html.* The $A_p$ and $f_{10.7}$ values for the day

prior to the simulation should be used in the model. The other two indices, $f_{hat}$ and $f_{400}$, are approximated by averaging the appropriate monthly mean values provided in the report. The recorded values for these indices are already in the correct units for entry into the model.

Next, the duration of the simulation and frequency of output are defined by four variables. $T_{int}$ is the start time for the simulation, entered in universal date/time format. *Freq* specifies how many times per orbit the loads will be calculated. The algorithm calculates the orbital period (in seconds) prior to setting these simulation parameters. Therefore, the analyst can use the period to specify a given time interval between calculations. For example, setting $freq = Per$ will tell the program to perform calculations once per second. The third simulation parameter, *dur* is the total number of orbital periods over which the simulation will run. Finally, *picnum* is equal to the number of points in the analysis for which graphic output will be displayed. These graphic displays will be created at even intervals throughout the duration of the simulation.

Once the orbital and simulation parameters are defined, the spacecraft model must be built. First, the material composition of the spacecraft must be specified. The required material properties are stored in a data structure called *Mat*. Each element of this data structure has the six fields shown in Table 4.5. The data structure must contain entries for each material found on the spacecraft. However, the structure is not limited to materials used in the current model. A database of commonly used spacecraft materials can be maintained here. The material index, k, is used in later routines to access the correct element of the material data structure.

Next, the complex satellite structure is broken into its basic components. For each component, a local coordinate system is defined. For example, the IAE model discussed in Chapter 2 and Appendix C was broken into six basic components: a rectangular box, three cylinders, a torus and a laminar disk. A local coordinate system was placed at the centroid of each component. While seemingly cumbersome,

Table 4.5    Material Data Structure

| Field | Contents |
|---|---|
| **mat(k).name** | the trade name of the material |
| **mat(k).dens** | mass density $(kg/m^3)$ |
| **mat(k).refl** | coefficient of reflection in the UV-visible spectrum |
| **mat(k).absp** | solar absorption coefficient |
| **mat(k).spec** | specular reflection coefficient |
| **mat(k).emm** | thermal emissivity coefficient |

this procedure of breaking the satellite into individual components greatly simplifies the process of defining node locations for complex structures. The components and their coordinate systems are defined in the data structure *comp*. Table 4.6 list the fields and contents of *comp*. The shape field is just a descriptor to help the

Table 4.6    Component Data Structure

| Field | Contents |
|---|---|
| **comp(i).shape** | description of the component |
| **comp(i).orgin** | location of local reference frame relative to the base frame |
| **comp(i).euler** | rotation of local reference frame relative to the base frame |
| **comp(i).type** | type of coordinate system used |

user distinguish between components. The algorithm will automatically take the first component defined and set its local reference frame as the base frame. All subsequent components' local reference frames will be described relative to the base frame. *comp.orgin* is a 3x1 vector containing the cartesian coordinates (x,y,z) of the local frame orgin relative to the base frame. *comp.euler* is a 3x1 vector containing the rotation angles required to transform from the base frame to the local reference

frame using an Euler 1-2-3 sequence. Finally the *type* field contains an integer that specifies which type of coordinates are used to describe nodes on that component. Valid entries include cartesian coordinates (1), cylindrical coordinates (2), spherical coordinates (3) and toroidal coordinates (4). Once each component is defined, the orientation of the base frame relative to the orbit frame must be specified. The orbit frame was defined in section 2.3 with the first axis along the satellite position vector and the third axis along the orbit normal. The base frame orientation is specified by the variables phi1, phi2 and phi3, the Euler 1-2-3 rotation angles. Once again, all angles are entered in degrees.

Now that the satellite structure has been broken down into its basic components, the finite element model can be defined. The first step is to specify locations for all of the nodes. This information will be in a data structure called *node*. Each element of *node* has two fields as shown in Table 4.7. *Node(j).comp* specifies which

Table 4.7    Node Data Structure

| Field | Contents |
|-------|----------|
| **Node(j).comp** | indicates which reference frame is used to describe the node |
| **Node(j).loc** | coordinates of the node in the specified frame |

component's local coordinate system is used to define the node's location. The entry in this field is just the index, i, for the appropriate element of the *comp* data structure. For nodes located at the junction of two components, either component's coordinate system can be used, whichever is more convenient. *Node(j).loc* is a column vector containing the node's coordinates in the specified coordinate system. The format of the node's coordinates depend on the type of coordinate system used. In cartesian coordinates, the node location is simply [x;y;z]. In cylindrical coordinates, the location is given by [r;$\theta$;z], where r is the radius of the cylinder and $\theta$ is the angle measured counterclockwise from the positive x axis to the projection of the radius vector into the x-y plane. Spherical coordinates are entered as [r,$\theta$, $\phi$]

where r is the radius of the sphere, $\theta$ is as defined above, and $\phi$ is the angle between the positive z axis and the radius vector. Cylindrical and spherical coordinates are displayed in Figure 4.1. Toroidal coordinates are entered as [R,$\theta$,r,$\phi$], as illustrated in Figure 4.2. R is the radius of curvature of the torus, while r is the radius of the torus cross section. $\theta$ is the clockwise measured angle between the positive x axis and R. $\phi$ is the angle measured clockwise from the positive z axis to r. No matter which coordinate system is used, all angles are given in degrees and distances are in meters.
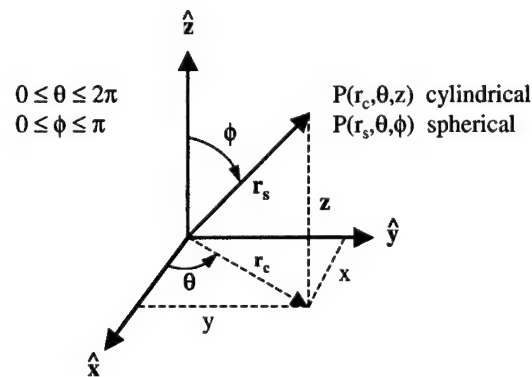


Figure 4.1    Cylindrical and Spherical Coordinates
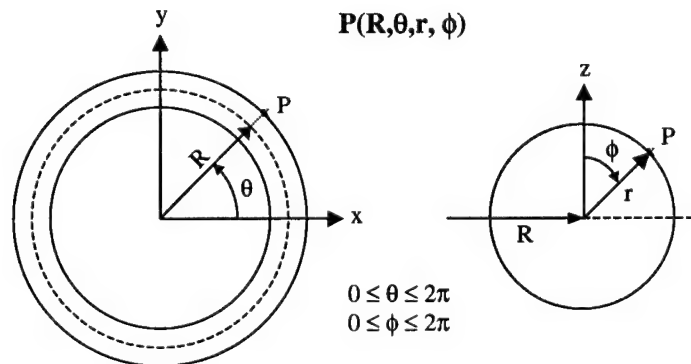


Figure 4.2    Toroidal Coordinates

Once the nodes are all defined, the elements are built. Element specifications are held in a data structure called *inelement*. Each entry contains the four fields listed in Table 4.8. As mentioned earlier, an element may be defined by one, two,

Table 4.8    Element Data Structure

| Field | Contents |
|---|---|
| **inelement(m).nodes** | specifies which nodes bind the element |
| **inelement(m).mat** | defines which material composes the element |
| **inelement(m).thick** | element thickness (meters) |
| **inelement(m).comp** | specifies which component the element is located on |

three or four nodes. The *.nodes* field is a row vector containing the index numbers, j, of the nodes which define the element. The *.mat* and *.comp* fields hold the index numbers $k$ and $i$ respectively to designate the appropriate elements of the material and component data structures. The *.thick* field is self explanatory for triangular and quadrilateral elements. For point masses and line masses, this field contains the element's total mass (kg).

## 4.3   Sample Output and Analysis

In addition to the load matrices already discussed, the simulation produces graphical displays of the loading distribution and text files containing node and element lists. In order to demonstrate these products and validate the disturbance model, a sample analysis was run. The model used for this analysis is illustrated in Figure 4.3. This barbell-like structure consists of three basic components, two spheres joined by a cylinder. Each sphere has a radius of two meters and is constructed from 75-micron mylar. The cylinder has a one-meter radius and length of four meters. It is constructed of 2mm Neoprene-coated Kevlar. These are the same materials used in the IAE model. The corresponding thermal and optical properties are listed in Table 2.1. The structure is oriented such that the $-b_3$ axis is always

pointed at the earth and the $-b_2$ axis is aligned with the orbit normal. The cylinder was divided into 1-meter segments along its length, and 30 deg arc segments around the circumference. Each of the spheres was discretized into 30 deg arc segments both horizontally and vertically. This created a finite element model consisting of 160 nodes defining 170 elements. The inputs for this model are listed in Appendix B.1.
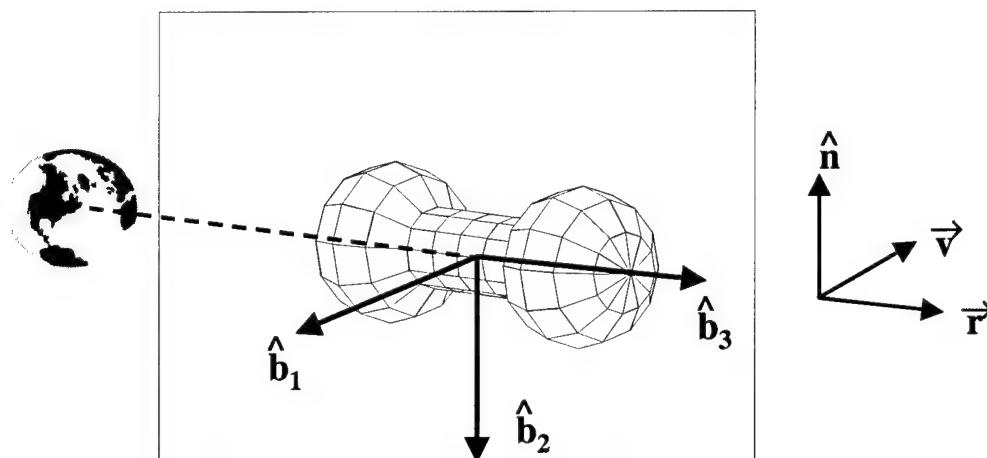


Figure 4.3    Sample Analysis Model

Whenever a simulation is run, the algorithm first executes the *structure* routine. Nodal coordinates are all transformed into a single body-fixed reference frame anchored at the vehicle center of mass. Elemental area, mass, orientation and material properties are calculated. At the end of this routine, two text files are output to the MATLAB work directory. *Node.txt* is a listing of all the nodes and their cartesian coordinates in the body-fixed frame. *Element.txt* is a listing of all the elements, along with several properties for each element. These properties include which component the element is located on, the surface area, mass, normal vector, material properties, location of the element centroid, and a list of the nodes which bind the element. The *Node* and *Element* files for the barbell model are included in Appendix B.2.

Three separate analyses were conducted with this model. First, the structure was put into circular orbit lying in the ecliptic plane at an altitude of 500km. The loads were checked at various positions within the orbit. The orbital position was characterized by $\eta$, the angle between the earth-sun line and the spacecraft position vector as defined in Figure 3.6. The simulation was started with the spacecraft lying on the earth-sun vector ($\eta = 0 \deg$). Loads were then calculated and graphically displayed at 90 deg intervals around the orbit. For the second analysis, $\eta$ was held constant at 270 deg and the orbital altitude was varied. Load calculations were performed at orbital altitudes of 300km, 1000km, 5000km and 20,183km (semi-synchronous orbit). All orbits in these two analyses were circular, lying in the ecliptic plane. A third simulation was run to check the variation of gravitational acceleration across the vehicle. This analysis utilized an eliptical orbit lying in the ecliptic plane. The results from all three simulations are summarized in Figures 4.4–4.8.

Figure 4.4 illustrates how the mechanical loads of gravity, drag and solar radiation pressure vary throughout the orbit. Because the orbit is circular, the $-b_1$ axis is always aligned with the velocity vector. Therefore, at each point in the orbit, the drag loading profile is the same. The elements oriented in the $-b_1$ direction experience the greatest drag. As the angle between an element's normal and the $-b_1$ axis increases, the drag load decreases. For all elements oriented at 90 deg or greater from this axis, the drag load is zero. Furthermore, since the orbit is circular the satellite altitude and velocity is constant for each point in the simulation. Therefore, the only change in drag is the variance in atmospheric density with latitude and longitude. This variance is too small to significantly alter the drag force. Therefore, the drag appears constant throughout the orbit. Similarly, the variance in gravitational loading throughout the orbit due to higher order terms in the earth's field is negligible compared to the overall load. Consequently, the gravity loading profile also appears constant throughout the orbit. The solar radiation pressure profile, in contrast, varies with position. When $\eta = 0 \deg$, the vehicle is directly between the

sun and earth, with the $b_3$ axis pointed toward the sun. The elements oriented in this direction show the greatest values for solar pressure, while elements facing away from this axis are shaded. At $\eta = 90$ deg, the vehicle is crossing the day-night terminator and the front of the structure $(-b_1)$ is shadowed. Likewise, when crossing back over the night-day terminator $(\eta = 270$ deg), the rear of the vehicle is shadowed and the front experiences the greatest solar radiation pressure. At $\eta = 180$ deg, the satellite is in umbral eclipse and no elements are subject to solar radiation pressure.

Figure 4.5 shows how the heat flux on each element from the various environmental sources varies throughout the orbit. As expected, the heating profile from direct solar radiation varies in the same manner as the solar pressure loading profile in the previous figure. The heating from earth-emitted radiation is constant throughout the circular orbit, with elements oriented toward the earth $(-b_3)$ experiencing the greatest heating. This also is expected. However, the results for earth-reflected radiation heating do not seem to make sense at first glance. The heating profile at $\eta = 0$ deg shows that the elements on the cylinder are experiencing greater heat flux than the spherical elements facing directly towards the earth. This is due to the fact that these elements have a greater surface area than the spherical elements and a higher absorption coefficient. The profile at umbral eclipse $(\eta = 180$ deg) shows zero heating from earth-reflected radiation as expected. But, the profiles at the day-night terminators also show zero heat flux. One would expect some earth-reflected radiation to still be reaching the vehicle at this point. This discrepancy is caused by the approximation used for the view factor between a planar element and the sunlit part of the earth as given in equation (3.65). The $\cos \eta$ term set the view factor equal to zero. In reality, some earth-reflected radiation is still reaching the vehicle at this point. However, even at this low altitude, the magnitude of this reflected radiation is two full orders of magnitude less than the direct solar radiation. Therefore, the view factor approximation does not significantly affect the accuracy of the results.

Drag (Pa)    Gravity (Pa)    Solar Pressure (Pa)
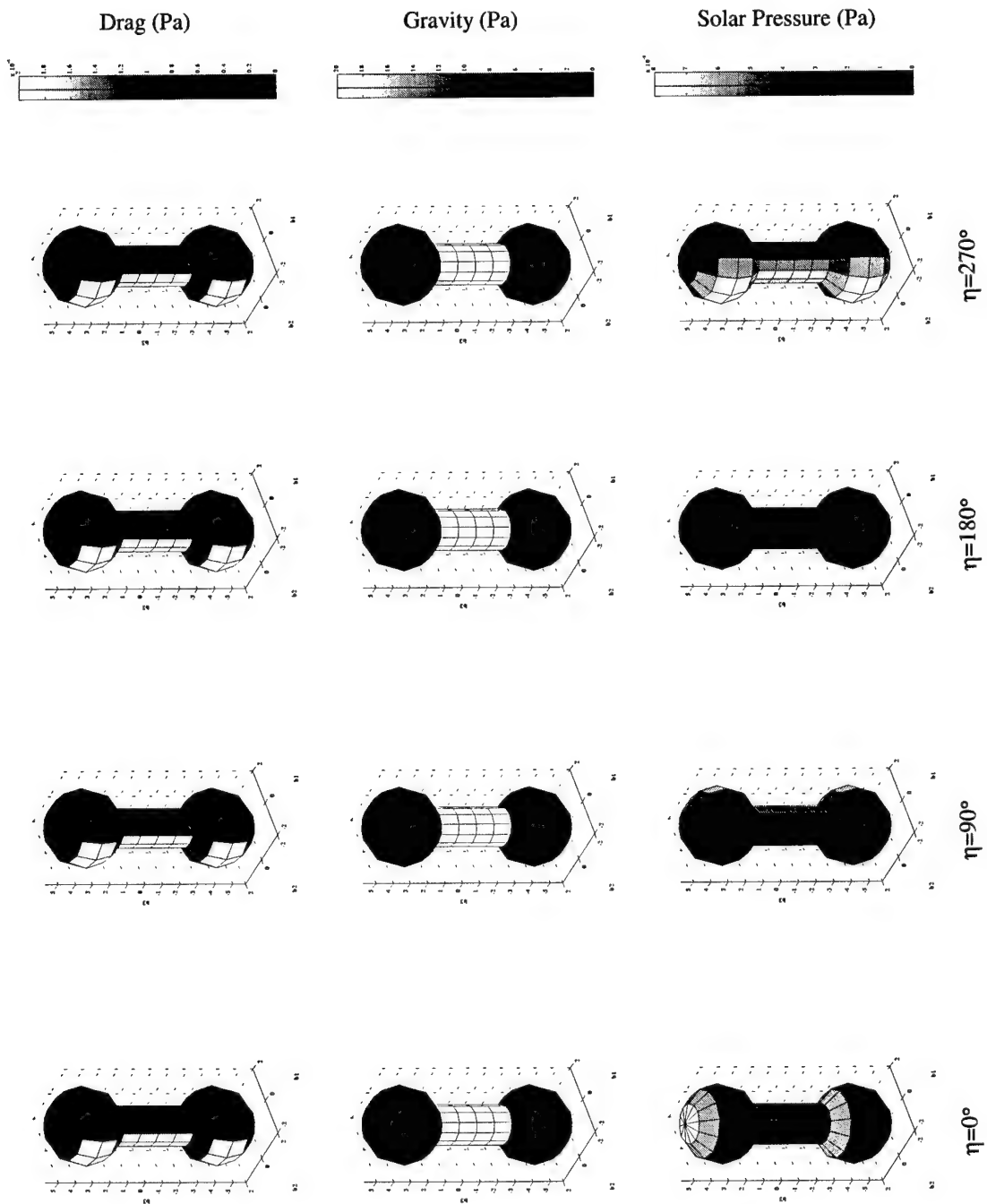


Figure 4.4    Mechanical Load vs. Orbital Motion, h=500km

Direct Solar (W)  Earth Reflected (W)  Earth Emitted (W)



$\eta=270°$

$\eta=180°$

$\eta=90°$

$\eta=0°$
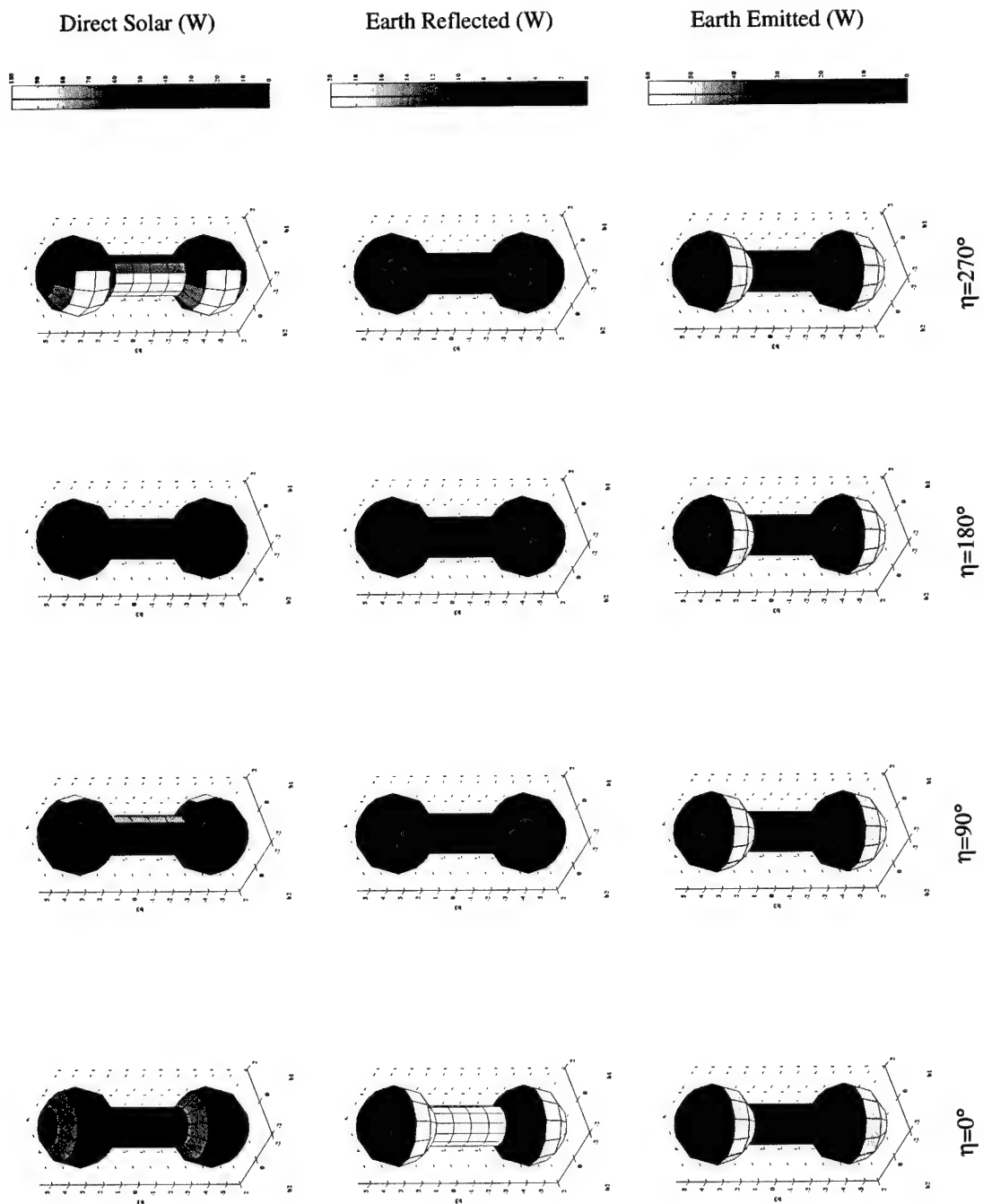
Figure 4.5    Heat Flux vs. Orbital Motion, h=500km

4-14

Figure 4.6 illustrates how the mechanical loads vary with orbital altitude. In order to facilitate easy comparison, each disturbance has been mapped against a common color scale. The numbers on the scales correspond to the base 10 log of the actual load. For example, a value of -3 indicates an actual load of $10^{-3}$ Pa. As expected, the solar radiation profiles are the same for each orbit, indicating that the solar pressure loading is independent of altitude. Also expected is the rapid decrease in drag loading from 300km to 1000km and the absence of drag for the two higher orbits. However, the gravity profile does not appear to conform to the predictions made in Chapter 2. The gravitational loading decreases with altitude, but not as rapidly as predicted by Figure 2.16. Also, the gravitational load is several orders of magnitude higher than drag or solar radiation pressure at all altitudes. According to Figure 2.16 drag should be the critical load in the 300km orbit, while solar radiation pressure should dominate in the semi-synchronous orbit.

This apparent inconsistency is easily explained by looking at what each figure measures. Figure 2.16 displays torque about the spacecraft center of mass as a function of altitude. Figure 4.6 shows how the gravity force on each element varies with orbital altitude. Recall that gravitational force is proportional to $1/R$, while gravity gradient torque is proportional to $1/R^3$. So, the values for gravity gradient should be much smaller. In this particular example, the total gravity gradient torque would actually be zero since the vehicle is in a stable orientation. (This is easily confirmed by calculating the torque about the vehicle center of mass caused by the gravity load on each element, and then summing across the entire structure.) Therefore, the results displayed in Figure 4.6 are reasonable.

Figure 4.7 shows how heat flux into each element varies with orbital altitude. As with Figure 4.6, each profile has been generated on a common scale to ease comparison between the various sources of heat flux. Also, the numbers on the scale represent the log of the actual thermal load. The profiles in this figure match the expected results. Direct solar radiation is the largest source of heat flux in
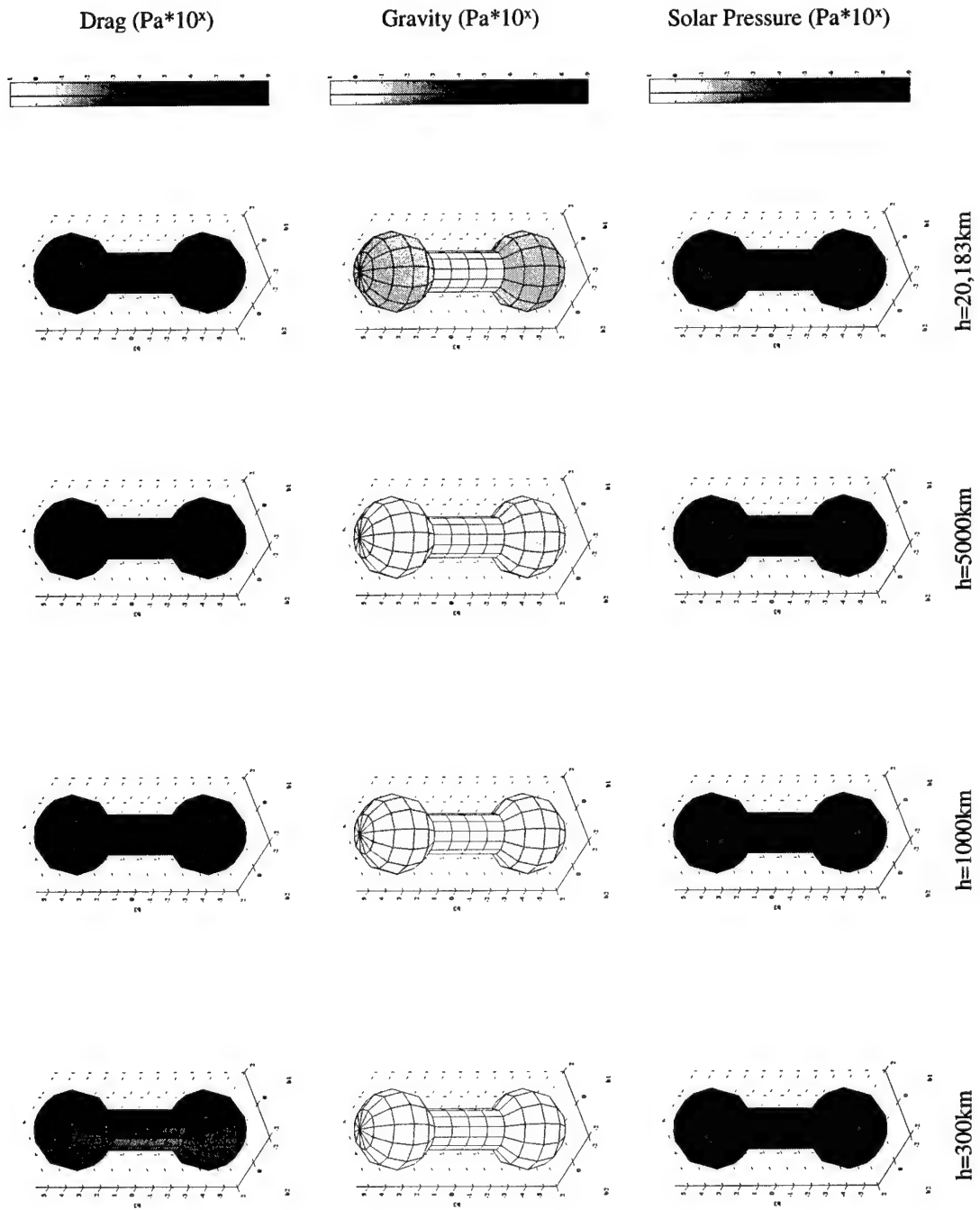
Figure 4.6    Mechanical Load vs. Orbital Altitude, $\eta = 270°$

Direct Solar (W*10ˣ)   Earth Reflected (W*10ˣ)   Earth Emitted (W*10ˣ)

h=20,183km

h=5000km

h=1000km

h=300km
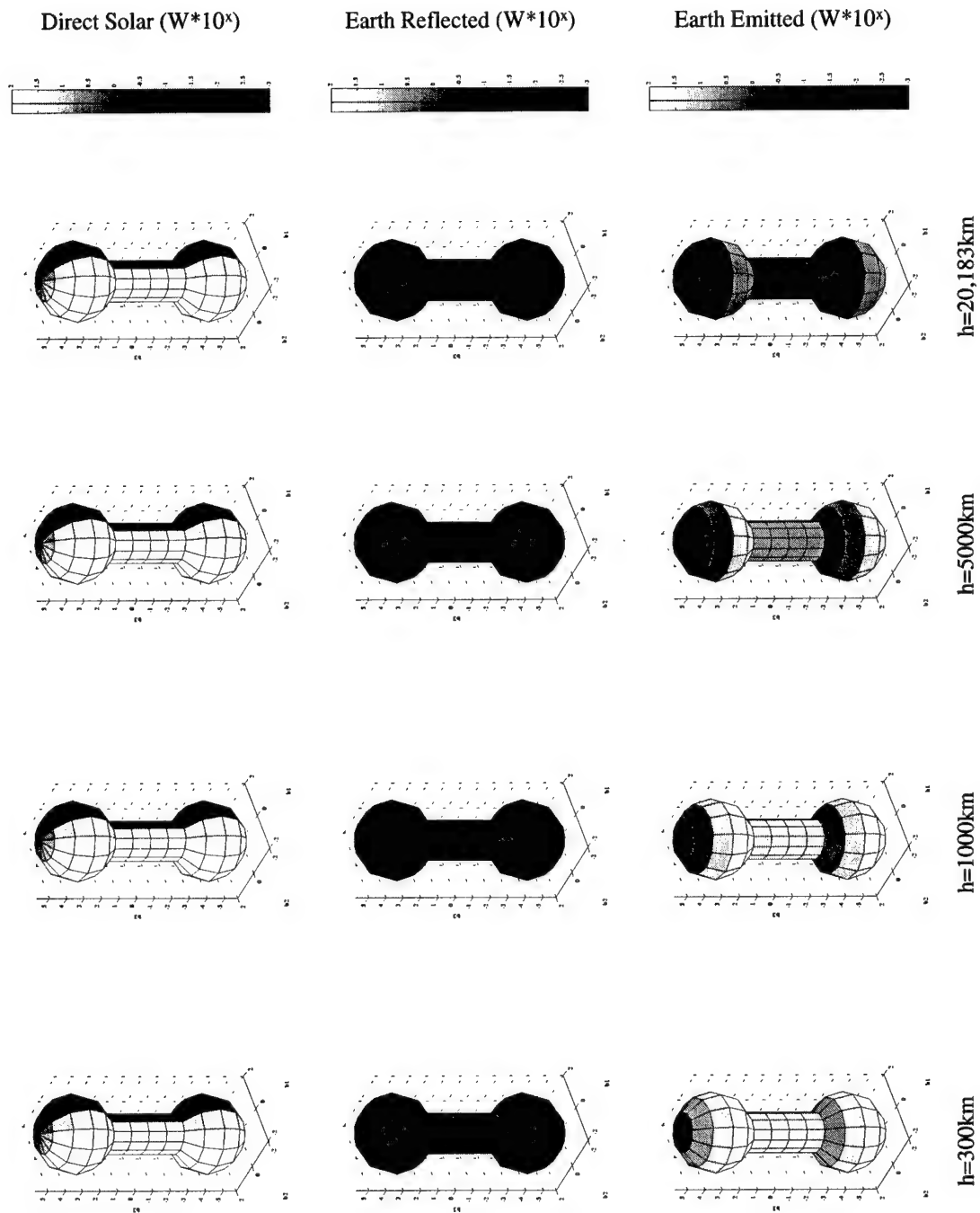
Figure 4.7    Heat Flux vs. Orbital Altitude, $\eta = 270°$

4-17

all of the orbits and is constant with respect to altitude. Earth-emitted radiation is a significant source of heat flux in the lower orbits, but drops off quickly with altitude. Earth-reflected radiation decreases with altitude similarly to earth-emitted radiation. However, at all altitudes it is 1-2 orders of magnitude less than earth-emitted radiation and 2-3 orders less than direct solar radiation.

Finally, Figure 4.8 displays how gravitational acceleration varies across the vehicle. The elliptical orbit used in this analysis had a semi-major axis of 7378 km and an eccentricity of 0.1. This causes the altitude to vary from 262 km at perigee to 1738 km at apogee. Perigee is located at the night-day terminator ($\eta = 270\,\text{deg}$). As expected, the gravitational acceleration varies with altitude, attaining a peak value at perigee of 9.0537 $m/s^2$ and a low of of 6.0575 $m/s^2$ at apogee. Furthermore, the variation of gravitational acceleration across the large distributed mass is graphically displayed. Stronger values of acceleration are experienced by the elements on the nadir-pointing end of the vehicle. Numerically, this variation is less than the four significant figures displayed on the colorbar scale. However, it is present and will be more pronounced for larger vehicles. This variation in gravitational acceleration will cause gravity gradient torques on satellites in non-equilibrium positions.

The results of these analyses conform to expectations based upon theory discussed in Chapter 2. Hand calculations for loads on selected elements in certain orientations confirm the values delivered by the code. This validates proper functioning of the program. Furthermore, the graphic output provides confirmation that the *structure* routine is correctly building the finite element model. The visual displays also allow the user to more easily analyze the loading profile experienced by the vehicle. Detailed graphic output for all analyses are included in Appendix B.2.

$\eta=12°$

$\eta=90°$

$\eta=168°$

$\eta=270°$
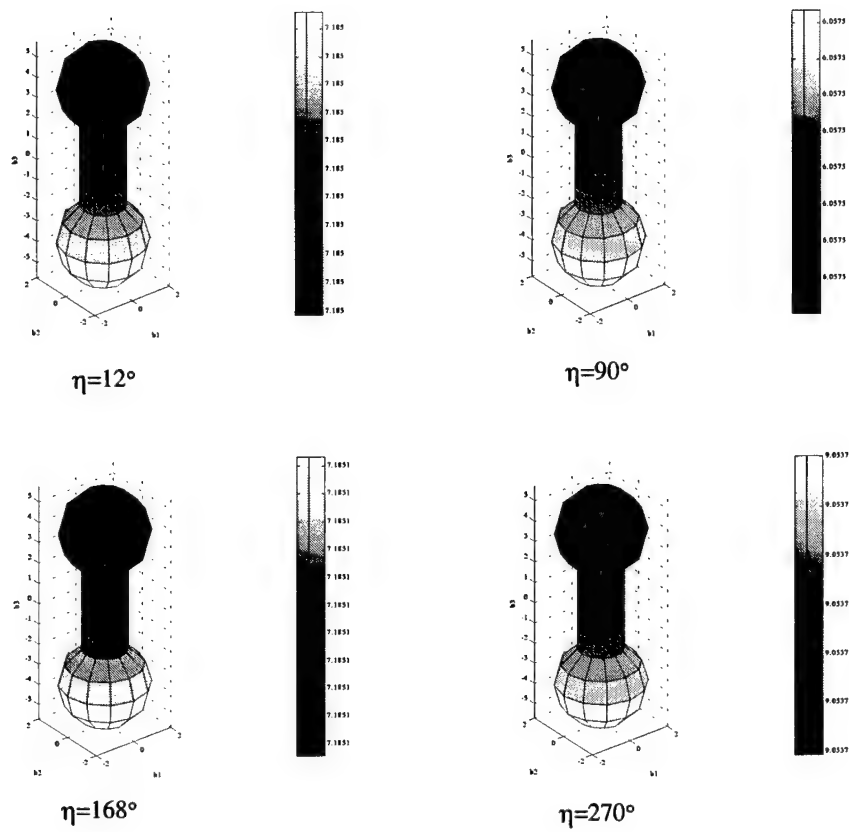
Figure 4.8    Gravitational Acceleration, $a = 7378 km, e = 0.1$

# V. Conclusions and Recommendations

## 5.1 Conclusions

The objective of this study was to determine the critical loads for large, inflatable space structures and then develop a model to calculate these loads for finite element analysis. The study focused on environmental disturbances, ignoring loads generated from within the spacecraft. Using the IAE as a base model, equations were developed to relate the torques generated by aerodynamic drag, gravity gradient, solar radiation pressure, thermal snap and magnetic disturbances to the satellite's orbital altitude. Since the generated torques are dependent on spacecraft attitude, a worst-case orientation was assumed in each analysis in order to make a fair comparison. The resultant torque versus altitude profiles are displayed in Figure 2.16. This plot identifies the critical loads for large space structures: gravity gradient, drag, solar radiation pressure, and thermal flux. The shape of these curves is almost identical to the torque profiles for small satellites shown in Figure 2.1. The difference occurs in the magnitude of the torques, which increases by a factor of $10^2$ for the larger satellites.

The one difference noted in the torque profiles for large versus small satellite was the shift of the magnetic torque curve relative to the other loads. Drag, gravity gradient, thermal and solar radiation torques increased by two orders of magnitude for the large structure; magnetic torque increased only by one order. This results from the fact that the other torques are directly related to the size of the structure, while the magnetic torque depends on the configuration of the current loops within the spacecraft. Depending on the size and orientation of these loops and the amount of current running through them, the shift in the magnetic torque curve can be either greater or less than the shift in the other loading curves. Therefore, care must be taken in designing these electrical circuits. However, this relationship also offers the potential for a form of attitude control. Since magnetic torque varies with altitude

in the same manner as gravity gradient, carefully controlling the orientation and current flow within various spacecraft circuits could help to counter gravity gradient torques.

The substantial increase in torques experienced by large satellites leads to another obvious conclusion; large inflatable structures have limited usefulness in low-earth orbits. The large torques generated at low altitudes will demand frequent attitude corrections. In particular, the significant drag force induced on large surface areas will hamper operations. Without recurrent energy-boosting maneuvers, the satellite orbit will quickly decay. Furthermore, the drag and gravity disturbances and the torques generated by the resulting attitude corrections will make it very difficult to maintain the dimensional accuracy of inflatable appendages. As mentioned previously, this is the primary design requirement for many of these structures. Therefore, missions utilizing inflatable structures should be designed for medium and high altitude orbits.

One caveat regarding Figure 2.16 must be mentioned. The torque due to thermal snap is modeled as a constant with respect to orbital altitude. This is based on a cylindrical earth shadow. In this model, there is no penumbral eclipse. No matter how high the orbit, the satellite transitions from umbral eclipse straight into full sunlight. This simplification was used to develop an expression relating temperature gradient to time. In reality, the earth shadow is more accurately modeled by the dual-conic configuration shown in Figure 3.5. As the orbit altitude increases, the satellite spends less time in umbra eclipse and more time in penumbra. As a result, the minimum temperature reached by sunward oriented surfaces will not be as low. Also, the preceding cool down and subsequent heating will be more gradual as less/more of the sun becomes visible during the penumbral transition. Therefore, changes in temperature gradients will also be more gradual. Since thermally-induced torques are proportional to the second time derivative of temperature gradient, the peak torques generated will diminish with altitude.

## 5.2 Recommendations for Further Research

With the critical loads identified, a disturbance model was generated. This code builds a faceted model as input by the user and then generates values for the critical loads at each specified time step. However, this only constitutes the first part of a finite element analysis. The end goal is to determine how these structures deform under the influence of orbital loads. Therefore, the loads must be input to another routine which then calculates the displacement of each node. Also, temperatures must be determined for each element. This will permit computation of the net heat flux and the resulting temperature gradients across satellite surfaces. With this information, the deformation of these inflatable structures can be predicted.

Furthermore, the disturbance model developed has room for improvement. First and foremost, a dynamics module should be added to the code. Currently, the satellite attitude relative to the local orbit frame is held constant throughout the analysis. In reality, torques on the satellite from these environmental disturbances will induce changing angular rates and displacements. The model already contains the code to calculate net torques about the body axes at each time step. Given these torques, adding a subroutine which integrates Euler's equations to update the satellite's attitude would be relatively simple and would increase the accuracy of the load calculations. Addition of an attitude control algorithm which determines the timing and magnitude of correction maneuvers based upon the updated satellite attitude would add even greater value to the model. According the Hedgepath [11], the torques generated by the environmental disturbances themselves are small compared to the torques caused by the corrections. Therefore, to truly appreciate the full impact of these environmental disturbances on the mission, the corresponding attitude control maneuvers must be analyzed.

The thermal analysis in the model offers another area for improvement. The routine predicts irradiance based only upon each element's orientation with respect to the sun and earth. In more complex structures, certain elements which face in the

direction of the sun may be shadowed by other surfaces. Also, radiation reflected from or transmitted through one element may be incident upon another element. These effects should be considered. This can be accomplished through the use of a ray tracing algorithm.

Finally, the disturbance model was designed with short term analyses in mind. Therefore, orbital perturbations were ignored. However, the environmental loads will change the orbital parameters over time. In fact, these orbital perturbation effects may be more pronounced than the structural distortions. Since the loads are dependent on orbital position and altitude, current orbital parameters must be input to the model. If longer term analyses are desired, an improved orbital propagator should be added to the code.

With continuing improvements in deployment precision and rigidization techniques, the Air Force and other space agencies are turning to inflatables for an increasing number of applications. This technology offers a relatively inexpensive and reliable method of deploying large structures in space using minimal lift capacity. Analytical tools such as this disturbance model provide a means of predicting the performance of these structures prior to actual deployment. This reduces the probablility of costly operational failures. While this disturbance model is not perfect, it represents an important first step toward predicting the behaviour of inflatable structures in orbit.

# Appendix A. Computer Code

## A.1 Forces.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%FORCES.M%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  ENVIRONMENTAL DISTURBANCE MODELLING FOR LARGE INFLATABLE STRUCTURES
%  CAPT DONALD J DAVIS, AFIT/ENY, GSO-01M
%
%  THIS PROGRAM CALCULATES THE ENVIRONMENTAL FORCES AND HEAT FLUXES EXPERIENCED BY
%  VARIOUS ELEMENTS OF A LARGE, FLEXIBLE STRUCTURE AS IT PROPAGATES THROUGH ITS
%  ORBIT IN ORDER TO RUN AN ANALYSIS, THE USER MUST CREATE A FILE MODEL.M WHICH
%  CONTAINS THE REQUIRED INPUTS FOR ANALYSIS TO INCLUDE: ORBITAL ELEMENTS, KEY
%  GEOMAGNETIC INDICES, SIMULATION PARAMETERS, MATERIAL PROPERTIES AND FINITE
%  ELEMENT MODEL DATA.  THE FILE NEWMODEL.M PROVIDES A TEMPLATE AND INSTRUCTIONS
%  FOR CREATING THE MODEL FILE.  GIVEN THIS INPUT, FORCES.M CALCULATES THE TOTAL
%  FORCE & HEAT FLUX ON EACH ELEMENT AT A GIVEN POINT IN TIME. KEY OUTPUTS INCLUDE
%
%  (for all below, j=# of elements in model, i=# of time steps in simulation
%
%        Time      i x 1 matix containing the time (JD format) at each look
%
%        dPg       3xjxi matix of the distributed gravity force on each element
%                  - for point mass returns total gravity force(N)
%                  - for line mass returns gravity force per unit length (N/m)
%                  - for planar elements, force per unit area (Pa)
%        dPsr      3xjxi matrix of the solar radiation pressure on each element,
%                  at each look point, (Pa)
%        dPd       3xjxi matrix of the distributed drag force on each element,
%                  at each look point,  (Pa)
%        dP        3xjxi matrix of the total distributed force on each element,
%                  at each look point,   (Pa)
%        dQs       jxi matix of the heat flux from direct solar radiation on each
%                  element, at each look point, (W)
%        dQer      jxi matrix of the heat flux from earth-reflected solar
%                  radiation on each element, at each look point, (W)
%        dQet      jxi matrix of the heat flux from earth-emitted thermal
%                  radiation on each element, at each look point, (W)
%        dQe       jxi matrix of the total heat flux from environmental sources
%                  on each element, at each look point, (W)
%
%  All of the forces are expressed in body-fixed coordinates.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clear all
global mu Re alb c Rs Ap f10 fhat f400
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%DEFINE CONSTANTS%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

mu=3.98601e14;                  %Earth gravitational parameter (m^3/sec^2)
Re=6378135;                     %Earth radius (m)
alb=0.36;                       %Earth albedo
c=3e8;                          %speed of light (m/s^2)
Rs=695508;                      %Sun radius (km)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%LOAD FINITE ELEMENT STRUCTURE
structure;
load structure;

%CONVERT ANGLES FROM DEGREES TO RADIANS
xi=xi*(pi/180);
argp=argp*(pi/180);
anode=anode*(pi/180);

a=a*1000;   %convert to meters

%CONVERT TIMES TO JULIAN DAYS
To=date2jd(To);
Tint=date2jd(Tint);

%TRANSFORMATION MATRIX FROM BODY FRAME TO ORBIT FRAME
Cba=[cos(phi3)*cos(phi2),-sin(phi3)*cos(phi2),sin(phi2);
   cos(phi3)*sin(phi2)*sin(phi1)+sin(phi3)*cos(phi1),...
      -sin(phi3)*sin(phi2)*sin(phi1)+cos(phi3)*cos(phi1),...
      -sin(phi1)*cos(phi2);
   -cos(phi3)*sin(phi2)*cos(phi1)+sin(phi3)*sin(phi1),...
      sin(phi3)*sin(phi2)*cos(phi1)+cos(phi3)*sin(phi1),...
      cos(phi2)*cos(phi1)];

%TRANSFORMATION MATRIX FROM PQW TO ECI FRAME
Cpi=[cos(anode)*cos(argp)-sin(anode)*cos(xi)*sin(argp),...
      -cos(anode)*sin(argp)-sin(anode)*cos(xi)*cos(argp),...
      sin(anode)*sin(xi);
   sin(anode)*cos(argp)+cos(anode)*cos(xi)*sin(argp),...
      -sin(anode)*sin(argp)+cos(anode)*cos(xi)*cos(argp),...
      -cos(anode)*sin(xi);
   sin(xi)*sin(argp),sin(xi)*cos(argp),cos(xi)];


%STEP SPACECRAFT THROUGH PERIOD OF INTEREST

looks=freq*dur;         %number of data points in one orbit
tstep=(Per/86400)/freq;

for i=1:looks
   Tep=Tint+(i-1)*tstep;
```

```
time(i,1)=Tep;
x=i

%CALCULATE R AND V FOR CENTER OF MASS (m,m/s,rad)
[Rcm(:,i),Vcm(:,i),ta]=randv(a,e,xi,argp,anode,To,Tep,Cpi);
Rnow=Rcm(:,i);
Vnow=Vcm(:,i);

%DETERMINE EARTH-SUN GEOMETRY (ECI)
svec=sunvec(Tep);              %Sun vector (km)
es=unit(svec);                 %Unit sun vector
des=mag(svec);                 %Earth-sun distance(km)
rdes=dot(Rnow,es)/mag(Rnow);
vdes=dot(Vnow,es)/mag(Vnow);
eta(i)=acos(rdes)*180/pi;      %Angle between es and Rnow
if vdes>0
   eta(i)=360-eta(i);
end

%DETERMINE SOLAR AND EARTH THERMAL RADIANCE (W/m^2)
Hsun=(3.8e26)/(4*pi*(des*1000)^2);
Het=Hsun*(1-alb)/4;

%CALCULATE TIME VARYING TRANSFORMATION MATRICIES
%body frame to ECI frame
Cap=[cos(ta),-sin(ta),0;sin(ta),cos(ta),0;0,0,1];
Cbi=Cpi*Cap*Cba;

%ECI to earth fixed frame
gmst=gmstime(Tep)*pi/180;    %Greenwich mean sidereal time (rad)
Cief=[cos(gmst),sin(gmst),0;-sin(gmst),cos(gmst),0;0,0,1];

%DETERMINE ATMOSPHERIC DENSITY
rho=density(Tep,Cief,Rnow);  %(kg/m^3)

for j=1:length(element)    %perform calculations for each element

   %FIND R FOR EACH ELEMENT(m) and ELEMENT NORMAL (ECI frame)
   rcmelb=element(j).centroid;%radius from cm to element in body frame
   rcmel=Cbi*rcmelb;           %radius from cm to element in ECI
   Rele=Rcm(:,i)+rcmel;        %radius from earth center to element
   norm=Cbi*element(j).norm;

   %ASSIGN MATERIAL PROPERTIES
   dm=element(j).mass;
   dA=element(j).area;
   beta=element(j).refl;
   del=element(j).spec;
   alpha=element(j).absp;
   emm=element(j).emm;
```

A-3

```
%FIND GRAVITATIONAL FORCE ON EACH ELEMENT (N)
dfg(:,j,i)=grav(Rele,Cief,dm,Cbi);    %body frame gravity force

%Find distributed gravity force (dPg)
if size(element(j).nodes,2)==1         %point mass
   dPg(:,j,i)=dfg(:,j,i);
elseif size(element(j).nodes,2)==2     %line mass
   lgth=mag(element(j).nodes(:,2)-element(j).nodes(:,1));
   dPg(:,j,i)=dfg(:,j,i)/lgth;
else                                   %planar elements
   dPg(:,j,i)=dfg(:,j,i)/dA;
end

%FIND SPACECRAFT-SUN VECTOR (ECI)
ss=svec-(Rele/1000);                   %S/C-sun vector (km)
rsvs=mag(ss);                          %S/C-sun distance (km)
ss=unit(ss);                           %S/C-Sun unit vector

%CRITICAL ANGLES FOR SOLAR RADIATION PRESSURE AND HEAT FLUX CALCULATIONS
rdotes=dot(Rele,es)/mag(Rele);    %radius vector dotted with earth-sun line
ndotss=dot(norm,ss);              %element normal dotted with S/C-sun line
ndotr=dot(-Rele,norm)/mag(Rele);  %element normal dotted with radius vector
rdotss=dot(-Rele,ss)/mag(Rele);   %s/c-earth vector dotted with s/c-sun line

%DETERMINE SOLAR INTENSITY(SI) FOR SOLAR PRESSURE AND HEAT FLUX CALCULATIONS
SI=solintens(Rele,rsvs,rdotss);

%FIND SOLAR RADIATION PRESSURE (SRP) ON EACH ELEMENT (N)
dPsr(:,j,i)=solpress(ss,norm,beta,del,SI,ndotss,Cbi,Hsun); %SRP
dfsr(:,j,i)=dPsr(:,j,i)*ndotss*dA; %net force on each element due to SRP

%CALCULATE HEAT FLUX ON EACH ELEMENT FROM ENVIRONMENTAL SOURCES (Watts)
[dQs(j,i),dQet(j,i),dQer(j,i)]=thermal(dA,emm,alpha,rdotes,ndotss,ndotr,...
   Rele,SI,Hsun,Het);

%CALCULATE THE DRAG DISTRIBUTED AND TOTAL FORCE ON EACH ELEMENT (N)
[dPd(:,j,i),dfd(:,j,i)]=drag(Vnow,norm,rho,dA,Cbi);

%TOTAL FORCE AND HEAT FLUX ON EACH ELEMENT
df(:,j,i)=dfd(:,j,i)+dfsr(:,j,i)+dfg(:,j,i); %Total force
dP(:,j,i)=dPd(:,j,i)+dPsr(:,j,i)+dPg(:,j,i); %Distributed force
dQe(j,i)=dQs(j,i)+dQet(j,i)+dQer(j,i);       %Heat flux

%CALCULATE TORQUES ABOUT SATELLITE CM FOR EACH ELEMENT(N-m)
%Tgge(:,j,i)=cross(rcmelb,dfg(:,j,i));     %gravity gradient
%Tsrpe(:,j,i)=cross(rcmelb,dfsr(:,j,i));   %solar radiation
%Tde(:,j,i)=cross(rcmelb,dfd(:,j,i));      %drag

end
```

```matlab
%CALCULATE TOTAL TORQUES ON THE SATELLITE AT EACH TIME STEP (N-m)
%Tgg(:,i)=sum(Tgge,2);                   %gravity gradient
%Tsrp(:,i)=sum(Tsrpe,2);                 %solar radiation
%Td(:,i)=sum(Tde,2);                     %drag
%Tt=Tgg+Tsrp+Td;                         %total torque

end


%PLOT MECHANICAL AND THERMAL LOAD ON EACH ELEMENT

for j=1:length(node)               %define vertices
   vert(j,:)=node(j).coord';
end

q=0;
for k=1:length(inelement)          %define faces
   if length(inelement(k).nodes)>2
      if length(inelement(k).nodes)==3
        q=q+1;
        fac(q,:)=[inelement(k).nodes NaN];
      else
         q=q+1;
         fac(q,:)=inelement(k).nodes;
      end
   end
end

%Display finite element model -no data
figure('Name','Model','NumberTitle','off','Color','w')
axis equal
axis vis3d
axis off
patch('Vertices',vert,'Faces',fac,'FaceColor','w')
view(3)
xlabel('b1')
ylabel('b2')
zlabel('b3')

%Display loading data
if picnum>0
   picstep=floor(looks/picnum);
   for i=picstep:picstep:looks
      q=0;
      for k=1:length(inelement)          %define color mapping schemes
         if length(inelement(k).nodes)>2
            q=q+1;
            mcolort(q,1)=mag(dP(:,k,i));
            mcolorg(q,1)=mag(dPg(:,k,i));
            mcolord(q,1)=mag(dPd(:,k,i));
            mcolorsr(q,1)=mag(dPsr(:,k,i));
```

```
            hcolort(q,1)=dQe(k,i);
            hcolors(q,1)=dQs(k,i);
            hcolorer(q,1)=dQer(k,i);
            hcoloret(q,1)=dQet(k,i);
        end
    end

    tag=num2str(eta(i));
    plotstring='graphic2';
    run(plotstring)
    end
end


save forces node element time dfg dfsr dfd df dPg dPsr dPd dP dQs dQer dQet dQe;
```

## A.2  Structure.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%STRUCTURE.M%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%FINITE ELEMENT STRUCTURE DEFINITION
%CAPT DON DAVIS, AFIT/ENY, GSO-01M
%
%THIS FILE TAKES THE FINITE ELEMENT MODEL DEFINED IN MODEL.M AND CALCULATES THE
%CENTROID AND SURFACE NORMAL FOR EACH ELEMENT AS WELL THE COMPOSITE BODY CENTER OF
%MASS.  THE ELEMENT NODES ARE REDEFINED IN TERMS OF THE BODY FIXED FRAME WITH ITS
%ORGIN AT THE CENTER OF MASS.  THE OUTPUT OF THIS FILE IS ELEMENT.*, A DATA
%STRUCTURE WITH LENGTH EQUAL TO THE NUMBER OF FINITE ELEMENTS IN THE MODEL.  EACH
%ELEMENT HAS THE FOLLOWING NINE FIELDS
%
%  element(i).nodes - node locations in body-fixed frame cartesian coordinates(m)
%  element(i).centroid - location of element centroid (m)
%  element(i).normal - outward normal for element surface
%  element(i).area - exposed element surface area (m^2)
%  element(i).mass - element mass (kg)
%  element(i).refl - coefficient of reflection
%  element(i).spec - coefficient of specular reflection
%  element(i).absp - solar absorptivity
%  element(i).emm - thermal emissivity
%
%THIS FILE ALSO OUTPUTS NODE AND ELEMENT SUMMARY DATA TO TWO TEXT FILES, NODE.TXT
%AND ELEMENT.TXT.  THESE FILES CAN BE FOUND IN THE WORK FOLDER OF THE MATLAB DIR.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Load input model
model;
load model;
```

```
%Convert angles from degrees to radians
phi1=phi1*(pi/180);
phi2=phi2*(pi/180);
phi3=phi3*(pi/180);

for i=1:length(comp)
    comp(i).euler=comp(i).euler*pi/180;
end

%REDEFINE ALL NODE IN BASE FRAME, CARTESIAN COORDINATES

for j=1:length(node)

    %Convert nodes to cartesian coordinates in local frame
    i=node(j).comp;      %i is the component index

    if comp(i).type==1
       tempnode=node(j).coord;

    elseif comp(i).type==2
       tempnode(1,1)=node(j).coord(1)*cos(node(j).coord(2)*pi/180);
       tempnode(2,1)=node(j).coord(1)*sin(node(j).coord(2)*pi/180);
       tempnode(3,1)=node(j).coord(3);

    elseif comp(i).type==3
       tempnode(1,1)=node(j).coord(1)*sin(node(j).coord(3)*pi/180)...
          *cos(node(j).coord(2)*pi/180);
       tempnode(2,1)=node(j).coord(1)*sin(node(j).coord(3)*pi/180)...
          *sin(node(j).coord(2)*pi/180);
       tempnode(3,1)=node(j).coord(1)*cos(node(j).coord(3)*pi/180);

    elseif comp(i).type==4
       tempnode(1,1)=(node(j).coord(1)+node(j).coord(3)...
          *sin(node(j).coord(4)*pi/180))*cos(node(j).coord(2)*pi/180);
       tempnode(2,1)=(node(j).coord(1)+node(j).coord(3)...
          *sin(node(j).coord(4)*pi/180))*sin(node(j).coord(2)*pi/180);
       tempnode(3,1)=node(j).coord(3)*cos(node(j).coord(4)*pi/180);

    else
      fprintf(1,'You did not specify a defined coordinate type for component %d',i)
    end

    %Define locent, a variable which holds the coordinates for the center of the
    %torus cross section at the node of interest.  This value is used later to
    %verify normal vectors for elements located on a torus.  For other shapes,
    %locent is set to [0;0;0]

    if comp(i).type==4
       node(j).loclent=[node(j).coord(1)*cos(node(j).coord(2)*pi/180);
          node(j).coord(1)*sin(node(j).coord(2)*pi/180);0];
```

```
    else
        node(j).loclent=[0;0;0];
    end


    %Calculate transformation matrix from local frame to base frame
    trans=[cos(comp(i).euler(2))*cos(comp(i).euler(3)),...
        -sin(comp(i).euler(3))*cos(comp(i).euler(2)),...
        sin(comp(i).euler(2));
     cos(comp(i).euler(3))*sin(comp(i).euler(2))*sin(comp(i).euler(1))...
        +sin(comp(i).euler(3))*cos(comp(i).euler(1)),...
        -sin(comp(i).euler(3))*sin(comp(i).euler(2))*sin(comp(i).euler(1))...
        +cos(comp(i).euler(3))*cos(comp(i).euler(1)),...
        -sin(comp(i).euler(1))*cos(comp(i).euler(2));
      -cos(comp(i).euler(3))*sin(comp(i).euler(2))*cos(comp(i).euler(1))...
        +sin(comp(i).euler(3))*sin(comp(i).euler(1)),...
        sin(comp(i).euler(3))*sin(comp(i).euler(2))*cos(comp(i).euler(1))...
        +cos(comp(i).euler(3))*sin(comp(i).euler(1)),...
        cos(comp(i).euler(2))*cos(comp(i).euler(1))];

    %Transform nodes and localc from local frame to base frame
    tempnode=trans*tempnode;
    node(j).loclent=trans*node(j).loclent;

    %Redefine node and localc position relative to orgin of base frame
    node(j).coord=tempnode+comp(i).orgin;

    if mag(node(j).loclent)>0.0
        node(j).loclent=node(j).loclent+comp(i).orgin;
    end

end %end node calculations

%BUILD ELEMENT DATA STRUCTURE

for m=1:length(inelement)

    i=inelement(m).comp;                %i is the component index

    %Enter node and cent coordinates
    for x=1:length(inelement(m).nodes)
        j=inelement(m).nodes(x);        %j is the node index
        temp(m).nodes(:,x)=node(j).coord;
        locent(:,x)=node(j).loclent;
    end

    %Collapse the locent matrix into a single reference point for each element
    for y=1:size(locent,2)
        cmag(y)=mag(locent(:,y));
    end
```

```matlab
[big,ind]=max(cmag);
if comp(i).type==4            %for elements on a torus, locent is set equal to
   locent=locent(:,ind);      %the center of the local torus cross section
else
   locent=comp(i).orgin;      %for elements on other shapes, locent is taken
end                           %to be the orgin of the local coordinate system


%Find element surface area, mass, centroid and normal

k=inelement(m).mat;       %k material index

if size(temp(m).nodes,2)==1               %point mass
   element(m).area=0;
   element(m).mass=inelement(m).thick;
   temp(m).centroid=temp(m).nodes;
   element(m).norm=[0;0;0];


elseif size(temp(m).nodes,2)==2           %line mass
   element(m).area=0;
   element(m).mass=inelement(m).thick;
   temp(m).centroid=temp(m).nodes(:,1)+0.5*(temp(m).nodes(:,2)...
      -temp(m).nodes(:,1));
   element(m).norm=[0;0;0];

elseif size(temp(m).nodes,2)==3           %triangular elements

   %Calculate area, mass and centroid
   base=temp(m).nodes(:,3)-temp(m).nodes(:,1);
   hyp=temp(m).nodes(:,2)-temp(m).nodes(:,3);
   theta=acos(dot(unit(base),unit(hyp)));
   height=mag(hyp)*sin(theta);
   midpt=temp(m).nodes(:,1)+0.5*base;
   med=temp(m).nodes(:,2)-midpt;
   element(m).area=0.5*mag(base)*height;
   element(m).mass=element(m).area*inelement(m).thick*mat(k).dens;
   temp(m).centroid=midpt+(1/3)*med;

   %Calculate normal
   tempnorm=cross(base,hyp);

   %Ensure normal is oriented outward
   localrad=temp(m).centroid-locent;
   rdotnorm=dot(localrad,tempnorm);
   if rdotnorm>0.0
      element(m).norm=unit(tempnorm);
   else
      element(m).norm=-unit(tempnorm);
   end
```

```matlab
    elseif size(temp(m).nodes,2)==4              %quadrilateral elements

        %Calculate area and mass
        base=temp(m).nodes(:,3)-temp(m).nodes(:,4);
        top=temp(m).nodes(:,2)-temp(m).nodes(:,1);
        hyp=temp(m).nodes(:,2)-temp(m).nodes(:,3);
        theta=acos(dot(unit(base),unit(hyp)));
        height=mag(hyp)*sin(theta);
        element(m).area=0.5*(mag(base)+mag(top))*height;
        element(m).mass=element(m).area*inelement(m).thick*mat(k).dens;

        %Calculate centroid
        med=temp(m).nodes(:,3)+0.5*(temp(m).nodes(:,1)-temp(m).nodes(:,3));
        centA=med+(1/3)*(temp(m).nodes(:,4)-med);
        centB=med+(1/3)*(temp(m).nodes(:,2)-med);
        massA=0.5*height*mag(base)*inelement(m).thick*mat(k).dens;
        massB=0.5*height*mag(top)*inelement(m).thick*mat(k).dens;
        temp(m).centroid=(1/element(m).mass)*(massA*centA+massB*centB);

        %Calculate normal
        tempnorm=cross(base,hyp);

        %Ensure normal is oriented outward
        localrad=temp(m).centroid-locent;
        rdotnorm=dot(localrad,tempnorm);
        if rdotnorm>0.0
            element(m).norm=unit(tempnorm);
        else
            element(m).norm=-unit(tempnorm);
        end

    else
        fprintf(1,'You did not specify an acceptable shape for element %d',m)
    end

    %Enter material properties into output data structure
    element(m).absp=mat(k).absp;
    element(m).emm=mat(k).emm;
    element(m).refl=mat(k).refl;
    element(m).spec=mat(k).spec;

end      %end element loop

%Calculate vehicle center of mass
massrad=[0;0;0];
for x=1:length(temp)
    massrad=massrad+element(x).mass*temp(x).centroid;
end
satcm=massrad/(sum([element.mass]));
```

```
%Redefine node and centroid locations relative to the vehicle cm
for k=1:length(element)
    element(k).centroid=temp(k).centroid-satcm;
    for m=1:size(temp(k).nodes,2)
        element(k).nodes(:,m)=temp(k).nodes(:,m)-satcm;
    end
end

for j=1:length(node)
    node(j).coord=node(j).coord-satcm;
end

%WRITE NODE DATA TO TEXT FILE
fid=fopen('node.txt','w'); fprintf(fid,'Node#
Coordinates(m)\r'); for k=1:length(node)
    fprintf(fid,'\n%3d       [%5.2f %5.2f %5.2f]\r',k,node(k).coord');
end
fclose(fid);

%WRITE ELEMENT DATA TO TEXT FILE
fid=fopen('element.txt','w');
fprintf(fid,'Element# Comp Area(m^2) Mass(kg)        Normal(m)       Absp    Emm'...
    '   Refl    Spec        Centroid(m)       Nodes\r');
for i=1:length(element)
    fprintf(fid,'\n %3d       %2d  %6.3f     %6.3f  [%5.1f %5.1f %5.1f]   %4.3f'...
        '   %4.3f    %4.3f    %4.3f  [%5.1f %5.1f %5.1f]',...
        i,inelement(i).comp,element(i).area,element(i).mass,element(i).norm',...
        element(i).absp,element(i).emm,element(i).refl,...
        element(i).spec,element(i).centroid');
    fprintf(fid,' (%d',inelement(i).nodes(1,1));
    if size(inelement(i).nodes,2)>1
        for j=2:size(inelement(i).nodes,2)
            fprintf(fid,',%d',inelement(i).nodes(j));
        end
    end
    fprintf(fid,')\r');
end
fclose(fid);

save structure;
```

## A.3  RandV.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [Rcm, Vcm, ta]=randv(a,e,xi,argp,anode,To,Tep,Cpi)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% RANDV FUNCTION FILE
% CAPT DONALD J DAVIS, AFIT/ENY, GSO-01M
%
% THIS FUNCTION TAKES A CLASSICAL ORBITAL ELEMENT SET AT A GIVEN TIME OF INTEREST
% AND CALCULATES POSITION AND VELOCITY VECTORS IN THE EARTH CENTERED INERTIAL
% (ECI) FRAME
%
% INPUTS:    a     semi-major axis (m)
%            e     eccentricity
%            xi    inclination (rad)
%            argp  argument of perigee (rad)
%            anode right ascension of the ascending node
%            To    time of perigee passage (modified julian day)
%            Tep   time of observation
%            Cpi   transformation matrix from perifocal to ECI frame
%
% OUTPUT:    Rcm   radius vector to satellite center of mass (ECI - m)
%            Vcm   velocity vector (ECI - m/s)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global mu Re alb c Ap f10 fhat f400

%CALCULATE MEAN ANOMALY (radians)
MA=mod(sqrt(mu/(a^3))*((Tep-To)*86400),2*pi);

%SOLVE KEPLER'S EQUATION TO FIND ECCENTRIC ANOMALY (radians)
EA=MA+e*sin(MA);
dE=1;

while abs(dE)>=0.00000001
   dm=EA-(e*sin(EA))-MA;
   dE=dm/(1-e*cos(EA));
   EA=EA-dE;
end

%FIND TRUE ANOMALY AND RADIUS (radians, m)
ta=2*atan(sqrt((1+e)/(1-e))*tan(EA/2));
rmag=(a*(1-e^2))/(1+e*cos(ta));

%POSITION AND VELOCITY OF CM IN PQW FRAME (m, m/sec)
rpqw=[rmag*cos(ta);rmag*sin(ta);0];
vpqw=sqrt(mu/(a*(1-e^2)))*[-sin(ta);e+cos(ta);0];

%POSITION AND VELOCITY OF CM IN ECI FRAME (m, m/sec)
Rcm=Cpi*rpqw;
Vcm=Cpi*vpqw;
```

A-12

## A.4 SunVec.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function svec=sunvec(jD)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SUNVEC FUNCTION FILE
% CAPT DONALD J DAVIS, AFIT/ENY, GSO-01M
%
% THIS FUNCTION CALCULATES EARTH-SUN VECTOR IN ECI COORDINATES FOR
% ANY GIVEN TIME.
%
% INPUTS:    jD       julian date
%
% OUTPUT:    svec     earth-sun vector (ECI) (km)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global mu Re alb c Rs Ap f10 fhat f400 Rpi

%Elapsed time since J2000
T=(jD-2451545.0)/36525;

%Obliquity of the ecliptic
obE=0.409092804-(2.26966e-4*T)-(2.86e-9*T^2)+(8.79e-9*T^3);

%Mean Anomaly
ma=6.240060141+(628.3019552*T)-(2.683e-6*T^2);
ma=mod(ma,2*pi);

%Eccentricity
ecc=0.016708634-(0.000042037*T)-(0.0000001267*T^2);

%Equation of center
C=(2*ecc-(0.25*ecc^3))*sin(ma)+(1.25*ecc^2)*sin(2*ma)+((13/12)*ecc^3)*sin(3*ma);

%True Anomaly
tas=mod(ma+C,2*pi);

%Earth-sun distance (km)
d=(1.000001018*(1-ecc^2)/(1+ecc*cos(tas)))*1.4959787e8;

%True longitude
tl=mod(4.938239961+tas,2*pi);

%Earth-sun vector
svec=d*[cos(tl);cos(obE)*sin(tl);sin(obE)*sin(tl)];
```

## A.5 Density.m

```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function rho=density(Tep,Cief,Rnow)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ATMOSPHERIC DENSITY FUNCTION FILE
% CAPT DONALD J DAVIS, AFIT/ENY, GSO-01M
%
% THIS FUNCTION CALCULATES THE LOCAL ATMOSPHERIC DENSITY ENCOUNTERED BY THE
% SATELLITE AS IT PROPAGATES THROUGH ITS ORBIT
%
% INPUTS:    Tep   current time (julian days)
%            Cief  ECI (inertial) to earth-fixed frame transformation matrix
%            Rcm   radius vector to the S/C center of mass
%
% OUTPUT:    rho   atmospheric density (kg/m^3)
%
% This file uses the function Atm70 from the Spacecraft Control Toolbox, designed
% by Princeton satellite systems.  This function calculates the atmospheric
% density using Jacchia's 1970 earth atmoshpere model for altitudes of 90 km to
% 2500 km.  For altitudes below 90 km, density is calculated using scale
% heights (AtmDens2-S/C control toolbox)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global mu Re alb c Rs Ap f10 fhat f400

%CALCULATE INPUT PARAMETERS FOR THE JACCHIA MODEL

%%days since Jan 1 of current year (djan1 - days)
Tepdt=jd2date(Tep);   %epoch time in DTG format [yr mon day hr min sec]
jan1dt=[Tepdt(1),1,1,Tepdt(4),Tepdt(5),Tepdt(6)];
jan1jd=date2jd(jan1dt);
djan1=Tep-jan1jd;

%%Latitude and longitude of satellite subpoint
Rcmf=Cief*Rnow;          %radius vector in earth-fixed frame
[lat,lon]=r2latlon(Rcmf);

GMT=(Tepdt(4)*60)+Tepdt(5)+(Tepdt(6)/60); %Minutes elapsed since 0000 GMT
alt=(mag(Rcmf)-Re)/1000;                  %Satellite altitude

%SET UP STRUCTURE FOR INPUT TO DENSITY MODEL FUNCTION
rhodata.aP=Ap;            %geomagnetic index 6.7 hours before computation
rhodata.dd=djan1;         %day number since Jan 1
rhodata.f=f10;            %daily 10.7 cm solar flux
rhodata.fHat=fhat;        %81 day mean of f10
rhodata.fHat400=f400;     %fhat 400 days before computation
rhodata.lat=lat;          %satellite latitude
rhodata.lng=lon;          %satellite longitude
rhodata.mm=GMT;           %minutes since 0000 GMT
rhodata.yr=Tepdt(1);      %current year
rhodata.z=alt;            %satellite altitude
```

```
%CALCULATE ATMOSPHERIC DENSITY (kg/m^3)
if alt<90              %Below 90 km use scale height model
   rho=AtmDens2(alt);
elseif alt>2500        %Above 2500 km - no atmosphere
   rho=0;
else                   %Use Jacchia model (90km-2500km)
   rho=AtmJ70(rhodata)*1000;
end
```

## A.6   Grav.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function dfg=grav(Rele,Cief,dm,Cbi)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% GRAVITY FUNCTION FILE
% CAPT DONALD J DAVIS, AFIT/ENY, GSO-01M
%
% THIS FUNCTION CALCULATES THE GRAVITATIONAL FORCE ON A SPACECRAFT AS IT
% TRAVELS THROUGH ITS ORBIT.  THIS FUNCTION USES THE SPACECRAFT CONTROL TOOLBOX
% FUNCTION AGRAVITY, WHICH CALCULATES GRAVITATIONAL ACCELERATION IN SPHERICAL
% COORDINATES USING NASA'S GEM-T1 MODEL.  THIS SPHERICAL HARMONIC MODEL OF
% EARTH'S GRAVITATIONAL FIELD IS COMPLETE TO DEGREE AND ORDER 36.
% INPUTS:    Rele    radius vector from earth to center of S/C element (m)
%            Cief    transform from GCI to earth fixed frame
%            dm      element mass (kg)
%            Cbi transformation matrix from body fixed to ECI frame
%
% OUTPUT:    dfg gravitational force on S/C element
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global mu Re alb c Rs Ap f10 fhat f400

%CONVERT RADIUS TO SPHERICAL COORDINATES(r,lambda,phi)

relef=Cief*Rele;               %r in earth fixed frame
[latg,long]=r2latlon(relef);
r=mag(relef)/1000;             %gravity model requires r in km
lambda=long;
phi=pi/2-latg;

%CALCULATE GRAVITATION ACCELERATION (m/s^2)
[ag,as,az,at]=agravity(6,6,r,lambda,phi);
Cspef=jsp2cart([1,lambda,phi]);%transform from spherical to cartesian(earth fixed)
ag=Cief'*Cspef*ag*1000;        %gravitational acceleration in ECI

dfg=Cbi'*(ag*dm);              %total drag force on element(body frame)
```

## A.7  SolIntense.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function SI=solintens(Rele,rsvs,rdotss)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SOLINTENS FUNCTION FILE
% CAPT DONALD J DAVIS, AFIT/ENY, GSO-01M
%
% THIS FUNCTION CALCULATES SOLAR RADIATION INTENSITY AS A FUNCTION OF THE
% SPACECRAFT'S POSITION IN ITS ORBIT.  INTENSITY CAN RANGE FROM 0 (UMBRAL ECLIPSE)
% TO 1(NO ECLIPSE).  DURING PENUMBRA, SOLAR INTENSITY WILL BE BETWEEN 0 AND 1.
% THIS INTENSITY WILL BE MULTIPLIED BY THE SOLAR RADIATION CONSTANT TO DETERMINE
% THE SOLAR RADIATION PRESSURE INCIDENT ON A SPACECRAFT ELEMENT.
%
% INPUTS:    Rele     radius vector from earth to center of S/C element (m)
%            rsvs     distance from S/C to sun (km)
%            rdotss   radius vector dotted with S/C-sun line
%
% OUTPUT:    SI       solar intensity (unitless, 0-1)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global mu Re alb c Rs Ap f10 fhat f400 Rpi

%DETERMINE S/C-EARTH-SUN GEOMETRY

betas=acos(rdotss);          %angle between s/c-sun line and s/c-earth line(rad)
rhos=asin(Rs/rsvs);          %angular radius of sun as viewed from s/c (rad)
rhoe=asin(Re/mag(Rele));     %angular radius of earth as viewed from s/c (rad)

%DETERMINE INTERFERENCE FACTOR

if betas>=(rhoe+rhos)        %no eclipse, SI=1
   f=0;
elseif betas<=(rhoe-rhos)    %umbral eclipse, SI=0
   f=pi*(rhos^2);
else                         %penumbral eclipse
   x=0.5*(rhos+rhoe+betas);
   h=(2/betas)*sqrt(x*(x-betas)*(x-rhos)*(x-rhoe));

   if (rhoe^2)-(rhos^2)<=betas^2
      f=(asin(h/rhos))*(rhos^2)+(asin(h/rhoe))*(rhoe^2)-h*betas;
   else
      f=(rhoe^2)*asin(h/rhoe)+(pi-asin(h/rhos))*(rhos^2)-h*betas;
   end
end

%CALCULATE SOLAR INTENSITY

SI=1-f/(pi*(rhos^2));
```

## A.8  SolPress.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function dPsr=solpress(ss,norm,beta,del,SI,ndotss,Cbi,Hsun)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SOLPRESS FUNCTION FILE
% CAPT DONALD J DAVIS, AFIT/ENY, GSO-01M
%
% THIS FUNCTION CALCULATES SOLAR RADIATION PRESSURE ON A SPACECRAFT AS IT
% PROPAGATES THROUGH ITS ORBIT
%
% INPUTS:    ss      unit vector pointing from the S/C element to the sun
%            norm    outward normal vector from S/C element
%            beta    coefficient of reflection (0-1)
%            del     percentage of reflected light which is reflected specularly
%            SI      solar intensity(0-1)
%            ndotss  cosine of angle between S/C normal and S/C-sun line
%            Cbi     transformation matrix from body fixed to ECI frame
%            Hsun    solar radiance at the orbit (W/m^2)
%
% OUTPUT:    dPsr    solar radiation pressure on S/C element (Pa)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global mu Re alb c Rs Ap f10 fhat f400

if ndotss<=0          %element is facing away from sun
   dPsr=[0;0;0];
else
   dPsr=-(Hsun/c)*SI*((1-del*beta)*ss+(2*beta*del*ndotss...
      +0.6666666667*beta*(1-del))*norm);
   dPsr=Cbi'*dPsr;  %express pressure in body frame
end
```

## A.9  Drag.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [dPd,dfd]=drag(Vnow,norm,rho,dA,Cbi)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DRAG FUNCTION FILE
% CAPT DONALD J DAVIS, AFIT/ENY, GSO-01M
%
% THIS FUNCTION CALCULATES THE TOTAL AND DISTRIBUTED DRAG FORCE ON A SPACECRAFT
% AS IT PROPAGATES THROUGH ITS ORBIT
%
% INPUTS:    Vnow    S/C velocity vector (ijk) (m/s)
%            norm    outward normal vector from S/C element (ijk)
%            rho     atmospheric density
%            dA      element area (m^2)
%            Cbi     transformation matrix from body fixed to ECI frame
```

```
% OUTPUT:   dPd   distributed drag force on S/C element
%           dfd   total drag force on S/C element
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global mu Re alb c Ap f10 fhat f400

vdotn=dot(Vnow,norm)/mag(Vnow);    %cosine of the angle between Vnow and norm

if vdotn<=0
   dPd=[0;0;0];
else
   %Calculate coefficient of drag
   thetad=acos(vdotn);
   Cd=2*(1+cos(2*thetad));

   dPd=-0.5*Cd*rho*mag(Vnow)*Vnow;  %distributed drag force
   dPd=Cbi'*dPd;                    %convert to body frame
end

dfd=dPd*dA*vdotn;       %total drag force in body frame
```

## A.10   Thermal.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [dQs,dQet,dQer]=thermal(dA,emm,alpha,rdotes,ndotss,ndotr,Rele,SI,Hsun,Het)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% THERMAL FUNCTION FILE
% CAPT DONALD J DAVIS, AFIT/ENY, GSO-01M
%
% THIS FUNCTION CALCULATES THE ENVIRONMENTAL HEAT FLUX INTO A SPACECRAFT
% AS IT PROPAGATES THROUGH ITS ORBIT
%
% INPUTS:    dA      area of spacecraft element (m^2)
%            emm     emmissivity of the element
%            alpha   absorbtivity of the element
%            rdotes  angle between S/C radius and earth-sun line
%            ndotss  angle between S/C normal and S/C-sun line
%            ndotr   angle between S/C normal and the radius vector
%            Rele    radius from earth center to S/C element (m)
%            SI      solar intensity (0-1)
%            Hsun    solar radiance (W/m^2)
%            Het     earth-thermal radiation (W/m^2)
%
% OUTPUT:    dQs     heat flux from direct solar radiation (Watts)
%            dQet    heat flux from earth thermal radiation  (W)
%            dQer    heat flux from earth reflected solar radiation (W)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
global mu Re alb c Rs Ap f10 fhat f400 Rpi

%CALCULATE HEAT FLUX FROM DIRECT SOLAR RADIATION (Watts)%%%%%%%%%%%%%%%%%%%%%%%%

if ndotss<=0        %element is facing away from sun
   dQs=0;
else
   dQs=alpha*Hsun*SI*dA*ndotss;
end

%CALCULATE HEAT FLUX FROM EARTH THERMAL RADIATION%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Find S/C element-earth view factor (Vet)
X=mag(Rele)/Re;
X2=X^2;
phi=asin(1/X);

if acos(ndotr)>pi/2+phi
   Vet=0;
elseif acos(ndotr)<pi/2-phi
   Vet=ndotr/X2;
else
   T1=(1/pi)*asin(sqrt(X2-1)/(X*sin(acos(ndotr))));
   T2=(1/(pi*X2))*(ndotr*acos(-sqrt(X2-1)*cot(acos(ndotr)))...
       -sqrt(X2-1)*sqrt(1-X2*(ndotr^2)));
   Vet=0.5-T1+T2;
end

dQet=emm*Vet*dA*Het;        %Heat flux from earth thermal radiation

%CALCULATE HEAT FLUX DUE TO EARTH-REFLECTED SOLAR RADIATION%%%%%%%%%%%%%%%%%%%%%

Ver=Vet*rdotes;     %config factor from S/C element to sunlit portion of earth

if Ver<0
   Ver=0;
else
   Ver=Ver;
end

dQer=alpha*alb*Ver*dA*Hsun; %earth reflected radiance
```

A-19

## A.11   Graphic2.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%GRAPHIC2.M%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% GRAPHICS FUNCTION FILE
% CAPT DONALD J DAVIS, AFIT/ENY, GSO-01M
%
% THIS FUNCTION PROVIDES THE GRAPHIC OUTPUT OF THE DISTRIBUTED FORCES ACROSS
% THE ENTIRE FINITE ELEMENT MODEL.  A 3D VIEW OF THE SATELLITE IS DISPLAYED,
% WITH EACH ELEMENT COLORED ACCORDING TO THE MAGNITUDE OF THE IMPOSED LOAD.
% TWO PLOTS ARE GENERATED AT EACH TIME STEP - ONE DISPLAYS THE MECHANICAL LOADS
% ON EACH ELEMENT WHILE THE OTHER SHOWS THE THERMAL LOADS.  THE TOP OF EACH
% FIGURE IS LABELED WITH SNAPSHOT TIME AND ETA, THE ANGLE BETWEEN THE EARTH-SUN
% LINE AND THE SPACECRAFT ECI POSITION VECTOR MEASURED IN THE DIRECTION OF MOTION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%PLOT THERMAL LOADS (W)
figure('Name',['Thermal Load(W): eta=',tag],'NumberTitle','off','Units',...
    'normalized','Position',[0.1,0.1,0.8,0.8]),clf
colormap(bone)

subplot(2,2,1)
patch('Vertices',vert,'Faces',fac,'FaceVertexCData',hcolors,'FaceColor','flat')
view(3)
axis equal
grid on
colorbar
xlabel('b1')
ylabel('b2')
zlabel('b3')
title('Direct Solar Radiation')

subplot(2,2,2)
patch('Vertices',vert,'Faces',fac,'FaceVertexCData',hcoloret,'FaceColor','flat')
view(3)
axis equal
grid on
colorbar
xlabel('b1')
ylabel('b2')
zlabel('b3')
title('Earth-Emitted Radiation')

subplot(2,2,3)
patch('Vertices',vert,'Faces',fac,'FaceVertexCData',hcolorer,'FaceColor','flat')
view(3)
axis equal
grid on
colorbar
xlabel('b1')
ylabel('b2')
zlabel('b3')
title('Earth-Reflected Radiation')
```

```
subplot(2,2,4)
patch('Vertices',vert,'Faces',fac,'FaceVertexCData',hcolort,'FaceColor','flat')
view(3)
axis equal
grid on
colorbar
xlabel('b1')
ylabel('b2')
zlabel('b3')
title('Total Heat Flux')

%PLOT MECHANICAL LOADS

figure('Name',['Mechanical Load(Pa): eta=',tag],'NumberTitle','off','Units',...
    'normalized','Position',[0.1,0.1,0.8,0.8]),clf
colormap(bone)

subplot(2,2,1)
patch('Vertices',vert,'Faces',fac,'FaceVertexCData',mcolorg,'FaceColor','flat')
view(3)
axis equal
grid on
colorbar
xlabel('b1')
ylabel('b2')
zlabel('b3')
title('Gravity')


subplot(2,2,2)
patch('Vertices',vert,'Faces',fac,'FaceVertexCData',mcolord,'FaceColor','flat')
view(3)
axis equal
grid on
colorbar
xlabel('b1')
ylabel('b2')
zlabel('b3')
title('Drag')

subplot(2,2,3)
patch('Vertices',vert,'Faces',fac,'FaceVertexCData',mcolorsr,'FaceColor','flat')
view(3)
axis equal
grid on
colorbar
xlabel('b1')
ylabel('b2')
zlabel('b3')
title('Solar Radiation Pressure')
```

```
subplot(2,2,4)
patch('Vertices',vert,'Faces',fac,'FaceVertexCData',mcolort,'FaceColor','flat')
view(3)
axis equal
grid on
colorbar
xlabel('b1')
ylabel('b2')
zlabel('b3')
title('Total Load')
```

## A.12  *Date2JD.m*

```
function jd = Date2JD( datetime )
%--------------------------------------------------------------------------------
%   Compute the Julian Date from the date. Uses the format from clock. If no
%   inputs are given, it will compute the Julian date for the instant
%   of the function call. Only works for dates after 1600.
%--------------------------------------------------------------------------------
%   Form:
%   jd = Date2JD( datetime )
%--------------------------------------------------------------------------------
%
%   ------
%   Inputs
%   ------
%   datetime               [ year month day hour minute seconds ]
%                          or the datetime data structure.
%                                  .year
%                                  .month
%                                  .day
%                                  .hour
%                                  .minute
%                                  .second
%
%
%   -------
%   Outputs
%   -------
%   jd                     Julian date
%
%--------------------------------------------------------------------------------

%--------------------------------------------------------------------------------
%   References: Montenbruck, O., T.Pfleger, Astronomy on the Personal
%               Computer, Springer-Verlag, Berlin, 1991, p. 12.
%--------------------------------------------------------------------------------
%   Copyright 1993 Princeton Satellite Systems, Inc. All rights reserved.
%--------------------------------------------------------------------------------

% Gives the current date if there are no inputs
%----------------------------------------------
if( nargin == 0 )
  datetime = clock;
else
  datetime = DTSToDTA( datetime );
end

if( datetime(2) == 0 ),
  error('No zero month')
end
```

```
% Adjust for negative years
%-------------------------
if( datetime(1) <= 0 )
  datetime(1) = datetime(1) + 1;
end


% datetime = [year month day hour minute second]
%-----------------------------------------------
fracday = (datetime(4) + (datetime(5) + datetime(6)/60)/60)/24;

a        = 1.e4*datetime(1) + 1.e2*datetime(2) + datetime(3) + fracday;

if( datetime(2) <= 2 )
  datetime(2) = datetime(2) + 12;
  datetime(1) = datetime(1) - 1;
end

if ( a <= 15821004.1 )
  b = -2 + fix((datetime(1) + 4716)/4) - 1179;
else
  b = fix(datetime(1)/400) - fix(datetime(1)/100) + fix(datetime(1)/4);
end

jd=365*datetime(1)+b+fix(30.6001*(datetime(2)+1))+datetime(3)+fracday+1720996.5;
```

## A.13   GMSTime.m

```
function gmst = GMSTime( jd )
%-------------------------------------------------------------------------------
%   Greenwich mean sidereal time - the angle between the Greenwich Meridian
%   and the Vernal Equinox
%-------------------------------------------------------------------------------
%   Form:
%   gmst = GMSTime( jd )
%-------------------------------------------------------------------------------
%
%   ------
%   Inputs
%   ------
%   jd                       Julian date UT (day)
%
%   -------
%   Outputs
%   -------
%   gmst                     Greemwich mean sidereal time (deg)
%
%-------------------------------------------------------------------------------

%-------------------------------------------------------------------------------
```

```
%    References: The Astronomical Almanac for the Year 1993, U.S. Government
%                Printing Office,1993, p. B6.
%-------------------------------------------------------------------------------
```

```
if( nargin == 0 )
  jd = Date2JD;
end

% Julian days at 0h UT
%---------------------
jd0h = R2P5( jd );

tu   = (jd0h - 2451545) / 36525;

% This organization maximizes the precision
%-------------------------------------------
gmst = (((0.093104 - 6.2e-6*tu).*tu + 8640184.812866).*tu + 24110.54841);

% Account for earth rotation
%---------------------------
gmst = gmst/86400  + (jd-jd0h)./MSidDay(jd);

% Limit to the range 0 to 360
%----------------------------
gmst = rem( gmst, 1 )*360;
```

## *A.14  JD2Date.m*

```
function datetime = JD2Date( jd, structOut )
%-------------------------------------------------------------------------------
%    Compute the calendar date from the Julian date. Uses the format
%    from clock. If no inputs are given it will output the current
%    date and time of the function call.
%-------------------------------------------------------------------------------
%    Form:
%    datetime = JD2Date( jd, structOut )
%-------------------------------------------------------------------------------
%
%    ------
%    Inputs
%    ------
%    jd                   Julian date
%    structOut            If entered, output a structure
```

```
%    -------
%    Outputs
%    -------
%    datetime                [ year month day hour minute seconds ]
%
%-------------------------------------------------------------------------------

%-------------------------------------------------------------------------------
%    References: Montenbruck, O., T.Pfleger, Astronomy on the Personal
%                Computer, Springer-Verlag, Berlin, 1991, p. 13.
%-------------------------------------------------------------------------------
%    Copyright 1993 Princeton Satellite Systems, Inc. All rights reserved.
%-------------------------------------------------------------------------------

if( nargin < 1 )
  jd = [];
end

if( isempty(jd) )
  datetime = clock;
else
  seconds = (jd-R2P5(jd))*86400;
  jd0      = fix(jd+0.5);

  if( jd0 < 2299161 ) % Gregorian calendar
    c = jd0;
  else
    b = fix(((jd0-1867216) - 0.25)/36524.25);
    c = jd0 + b - fix(b/4) + 1;
  end

  c = c + 1524;
  d = fix((c-122.1)/365.25);
  e = 365*d + fix(d/4);
  f = fix((c-e)/30.6001);
  datetime(2) = f-1-12*fix(f/14) ;
  datetime(1) = d-4715-fix((7 + datetime(2))/10);
  datetime(3) = fix(c-e+0.5)-fix(30.6001*f);
  datetime(4) = fix(seconds/3600);
  seconds      = seconds - 3600*datetime(4);
  datetime(5) = fix(seconds/60);
  datetime(6) = seconds - 60*datetime(5);

  if ( datetime(1) <= 0 ),
    datetime(1) = datetime(1)-1;
  end
end

if( nargin > 1 )
  datetime = DTAToDTS( datetime );
end
```

## A.15  R2LatLon.m

```
function [lat, lon] = R2LatLon( x, y, z )
%------------------------------------------------------------------------
%   Computes geocentric latitude and longitude from r
%------------------------------------------------------------------------
%   Form:
%   [lat, lon] = R2LatLon( x, y, z )
%   [lat, lon] = R2LatLon( r )
%------------------------------------------------------------------------
%
%   ------
%   Inputs
%   ------
%   x           (1,:)     X or [x;y;z]
%   y           (1,:)     Y
%   z           (1,:)     Z
%
%   -------
%   Outputs
%   -------
%   lat         (1,:)     Latitude (rad)
%   lon         (1,:)     East longitude (0 in xz-plane, +right hand rule
%                         about +z)  (rad)
%
%------------------------------------------------------------------------

%------------------------------------------------------------------------
%   Copyright 1993 Princeton Satellite Systems, Inc. All rights reserved.
%------------------------------------------------------------------------

if( nargin == 3 )
  r = [x;y;z];
else
  r = x;
end

u    = Unit(r);
lon  = atan2( u(2,:), u(1,:) );
latX = asin( u(3,:) );

if( nargout == 0 )
   Plot2D(lon*180/pi,latX*180/pi,'Longitude (deg)','Latitude (deg)',...
       'Latitude vs. Longitude');
else
  lat = latX;
end
```

A-27

## A.16 AtmDens2.m

```
function rhoOut = AtmDens2( h )
%-----------------------------------------------------------------------------
%   Computes the atmospheric density using scale heights.
%-----------------------------------------------------------------------------
%   Form:
%   rhoOut = AtmDens2( h )
%-----------------------------------------------------------------------------
%
%   ------
%   Inputs
%   ------
%   h                       Altitude (km)
%
%   -------
%   Outputs
%   -------
%   rhoOut                  Atmospheric Density (kg/m^3)
%
%-----------------------------------------------------------------------------

%-----------------------------------------------------------------------------
%   References:   Wertz, J.R., Spacecraft Attitude Determination and Control,
%                 Kluwer, 1976, p. 820.
%-----------------------------------------------------------------------------
%       Copyright 1994 Princeton Satellite Systems, Inc. All rights reserved.
%-----------------------------------------------------------------------------

rhoRef   = [1.225      3.899e-2  1.774e-2  8.279e-3  3.972e-3  1.995e-3...
            1.057e-3  5.821e-4  3.206e-4  1.718e-4  8.770e-5  4.178e-5...
            1.905e-5  8.337e-6  3.396e-6  1.343e-6  5.297e-7  9.661e-8...
            2.438e-8  8.484e-9  3.845e-9  2.070e-9  1.244e-9  5.464e-10...
            2.789e-10 7.248e-11 2.418e-11 9.158e-12 3.725e-12 1.585e-12...
            6.967e-13 1.454e-13 3.614e-14 1.170e-14 5.245e-15 3.019e-15];

hRef     = [ 0, linspace( 25,100,16), linspace(110, 160,6), 180,...
             linspace(200,500, 7), linspace(600,1000,5)];

lh       = length(hRef);

hScale   = (hRef(1:lh-1)-hRef(2:lh))./log(rhoRef(2:lh)./rhoRef(1:lh-1));
hScale   = [hScale hScale(length(hScale))];

if nargin > 0,
  for i = 1:length(h),
    j   = min(find(hRef>h(i)));
     if length(j) > 0,
        rho(i)         = rhoRef(j-1)*exp(-(h(i)-hRef(j-1))/hScale(j-1));
        rhoRef(1:j-2) = [];
        hRef  (1:j-2) = [];
        hScale(1:j-2) = [];
```

```
      else
        lh              = length(hScale);
        rho(i)          = rhoRef(lh)*exp(-(h(i)-hRef(lh))/hScale(lh));
      end
   end
end

if nargout == 0,
  NewFig('Atmospheric Density');
  if nargin == 0,
    semilogy(hRef,rhoRef)
  else
    semilogy(h,rho)
  end
  XLabelS('Altitude (km)')
  YLabelS('Density (kg/m^3)');
  TitleS('Atmospheric Density')
  grid
else
  [r,c] = size(h);
  if( r > c )
    rho = rho';
  end
  rhoOut = rho;
end
```

## A.17   AtmJ70.m

```
function [rho, nHe, nN2, nO2, nO, tZ] = AtmJ70( d )
%----------------------------------------------------------------------------------
%   Computes the atmospheric density using Jacchia's 1970 model.
%----------------------------------------------------------------------------------
%   Form:
%   rho = AtmJ70( d )
%----------------------------------------------------------------------------------
%   ------
%   Inputs
%   ------
%   d      (:) Data structure
%                .aP       Geomagnetic index 6.7 hours before the computation
%                .dd       Day number since Jan 1., days
%                .f        Daily 10.7 cm solar flux (e-22 watts/m^2/cycle/sec)
%                .fHat     81-day mean of f (e-22 watts/m^2/cycle/sec)
%                .fHat400  fHat 400 days before computation date
%                .lat      Latitude of computation point + north (deg)
%                .lng      Longitude of computation point + east (deg)
%                .mm       Greenwich mean time from 0000 GMT, minutes
%                .yr       Year
%                .z        Geometric altitude (km)
```

```
%    -------
%    Outputs
%    -------
%    rho  (:) Density (g/cm^3)
%
%---------------------------------------------------------------------------


%---------------------------------------------------------------------------
%    Reference: Models of the Earth's Atmosphere (90 to 2500 kM) NASA SP-8021.
%
%                Roberts, C.E. Jr, "An Analytic Model for Upper Atmosphere
%                Densities Based Upon Jacchia's 1970 Models", Celestial Mechanics
%                Vol. 4, 1971, pp. 368-377.
%---------------------------------------------------------------------------
%    Copyright 1999 Princeton Satellite Systems, Inc.
%    All rights reserved.
%---------------------------------------------------------------------------

ang = 23.45;
aV  = 6.02257e23; % Avogadro's number (per mole)

mH  = 1.00797;    % atomic mass of hydrogen
mN2 = 2*14.0067;  % molecular mass of diatomic nitrogen
mO2 = 2*15.9994;  % molecular mass of diatomic oxygen
mO  = 15.9994;    % atomic mass of oxygen
mHe = 4.00260;    % atomic mass of helium

wH  = 1.6731e-24; % hydrogen mass (g/molecule)
wHe = 6.6435e-24; % helium   mass (g/molecule)
wN2 = 4.6496e-23; % nitrogen mass (g/molecule)
wO2 = 5.3104e-23; % diatomic oxygen mass (g/molecule)
wO  = 2.6552e-23; % oxygen   mass (g/molecule)

qN2 = 0.78110;    % low atmosphere volumetric fraction of N2
qAr = 0.00934;    % low atmosphere volumetric fraction of argon
qHe = 1.289e-5;   % low atmosphere volumetric fraction of helium
qO2 = 0.20955;    % low atmosphere volumetric fraction of O2

z90   = 90;             % 90km reference altitude (km)
rho90 = 3.46e-9;  % assumed density at 90km altitude (g/cm^3)
t90   = 183;      % assumed temperature at 90km altitude (deg-K)
m90   = 28.878;   % assumed molecular mass at 90km altitude (unitless)

% NASA Equations
%---------------
j      = 2441683 + (d.yr - 1973)*365 + d.dd; % (A-1)
jStar  = (j - 2415020)/36525; % (A-2)

gP     = 99.6909833 + (36000.76854 + 0.00038708*jStar).*jStar...
         + 0.25068447*d.mm; % (A-3)
```

```matlab
rAP     = Range( gP + d.lng, 0, 360 ); % (A-4)


dJ      = j - 2435839;
lS      = Range((0.017203*dJ + 0.0335*sin( 0.017203*dJ ) - 1.41)...
   *180/pi, -180, 180); % (A-5)


dS      = ASinD( SinD( lS )*SinD( ang ) ); % (A-6)
rASArg = TanD( dS )/TanD( ang );

% Put rAS in the same quadrant as lS
%-----------------------------------
rAS     = ASinDSameQuadrant( rASArg, lS );
rAS     = Range( rAS, 0, 360 );


hRA     = rAP - rAS; % (A-8) this should be >0

% Angle between bulge and computation point
%-------------------------------------------
tau     = Range( hRA - 37 + 6*SinD( hRA + 43 ), -180, 180 ); % (A-9)

% Nightime minimum global exospheric temperature (deg-K)
%-------------------------------------------------------
tC      = 383 + 3.32*d.fHat + 1.8*(d.f - d.fHat ); % (A-10)

% Diurnal correction (deg-K)
%---------------------------
eta     = 0.5*abs(d.lat - dS);
theta   = 0.5*abs(d.lat + dS);
r       = -0.19 + 0.25*log10(d.fHat400);

z       = SinD(theta)^2.5;

a       = r.*(( CosD(eta)^2.5 - z )./( 1 + r.*z ));
tL      = tC.*(1 + r.*z).*(1 + a.*CosD(0.5*tau)^3); % (A-11)

% Geomagnetic activity correction (deg-K)
%----------------------------------------
tG      = d.aP + 100*(1 - exp(-0.08*d.aP)); % (A-12)

% Semiannual correction (deg-K)
%------------------------------
z       = d.dd/365.2422;
tau     = z + 0.1145*( (0.5*(1+SinD(360*z+342.3))).^2.16 - 0.5 );
tS      = 2.41 + d.fHat.*(0.349 + 0.206*SinD(360*tau+226.5))...
   *SinD(720*tau+247.6); % (A-13)

% Exospheric temperature (deg-K)
%-------------------------------
tE      = tL + tG + tS; % (A-14)
```

```matlab
% Inflection point temperature (deg-K at altitude = 125 km)
%-----------------------------------------------------------
tX     = 444.3807 + 0.02385*tE - 392.8292*exp(-0.0021357*tE); % (A-15)

% Temperature at geometric altitude levels (deg-K)
%-------------------------------------------------
dZ      = d.z - 125;

l       = find(dZ <= 0 );
if( ~isempty(l) )   % Altitudes between 90 and 125 km
    tZ(l)  = TempLowAlt( d.z(l), tX, t90 );
end;

l       = find(dZ > 0 );
if( ~isempty(l) )
    tZ(l) = TempHighAlt( d.z(l), tX, t90, tE );
end;

% For altitude <= 105 km
%-----------------------
l = find(d.z <= 105 );

if( ~isempty(l) )

  % Mean molecular mass (unitless)
  %-------------------------------
    eM(l) = MolMassLowAlt( d.z(l) );

  % Mass density before seasonal-latitudinal correction
    %   Integrate barometric equation (A-19)
  %------------------------------------------------------
  baromInt = quad( 'BaromExp', 90, d.z(l), [], [], tX, t90 );
    rho(l)  = rho90 * (t90/tZ(l)) * (eM/m90) * exp( baromInt );

  % Seasonal-latitudinal density correction
  %-----------------------------------------
  dZ      = d.z(l)-90;
  dDD     = 0.02*dZ*exp(-0.045*dZ)*SinD(360/365.2422*(d.dd(l)+100))...
      *(SinD(d.lat(l))).^2 .* sign(d.lat(l)); % (A-20)
  rho(l) = rho(l)*10^dDD;

  % Number densities
  %-----------------
  par(l) = aV*rho(l)/eM(l);          % total number of particles per cm^3
  nN2(l) = qN2*eM(l).*par(l)/28.96;
  nHe(l) = qHe*eM(l).*par(l)/28.96;
  nAr(l) = qAr*eM(l).*par(l)/28.96;
  n02(l) = par(l).*(eM(l)*(1+q02)/28.96-1);
  n0(l)  = 2*par(l).*(1-eM(l)/28.96);

end;
```

```
% Must calculate parameters at 105 km
%----------------------------------
tZ105  = TempLowAlt( 105, tX, t90 );
eM105  = MolMassLowAlt( 105 );

baromInt = quad( 'BaromExp', 90, 105, [], [], tX, t90 );
rho105   = rho90 * (t90/tZ105) * (eM105/m90) * exp( baromInt );

dZ       = 105-90;

dDD105 = 0.02*dZ*exp(-0.045*dZ)*SinD(360/365.2422*(d.dd+100))...
   *(SinD(d.lat)).^2 .* sign(d.lat); % (A-20)
rho105 = rho105*10^dDD105;

par105 = aV*rho105/eM105; % (A-22)

% Molecular number densities at 105 km
%------------------------------------
fac    = eM105/28.96;
nN2105 = qN2*par105*fac;          % (A-23)
nHe105 = qHe*par105*fac;          % (A-23)
nO2105 = par105*(fac*(1+qO2)-1);  % (A-24)
nO105  = 2*par105*(1-fac);        % (A-25)



% For altitude > 105 km
%---------------------
l = find(d.z > 105 );

if( ~isempty(l) )

  diffusionInt  = quad( 'DiffusionExp', 105, d.z(l), [], [], tX, t90, tE );
  tR            = tZ105./tZ(l);

  % N2, O2, O, He number density (cm^-3)
  %------------------------------------
  nN2   = nN2105 * tR * exp( mN2*diffusionInt ); % (A-28)
  nO2   = nO2105 * tR * exp( mO2*diffusionInt ); % (A-28)
  nO    = nO105  * tR * exp( mO *diffusionInt ); % (A-28)
  nHe   = nHe105 * tR^0.62 * exp( mHe*diffusionInt); % (A-28)

  % Hydrogen number density (cm^-3)
  %-------------------------------
  nH    = zeros(1,length(l));
  l2    = find( d.z(l) > 500 );
  if( ~isempty(l2) )
    tZ500   = TempHighAlt( 500, tX, t90, tE );
    lTE     = log10(tE);
    diffusionInt = quad( 'DiffusionExp', 500, d.z(l(l2)), [], [], tX, t90, tE );
```

```
    nH500   = 10^( 73.13 - ( 39.4 - 5.5*lTE)*lTE );   % (A-26)
    nH(12)  = nH500 .* ( tZ500./tZ(1(12)) ) * exp( mH*diffusionInt );   % (A-27)
  end;

  % Seasonal-latitudinal variation of helium
  %----------------------------------------
  dHe  = 0.5 + 1.8*( ((23.45-dS(1))/47.5)^2.5 * (SinD(45+d.lat(1)/2))^4 ...
          + ((23.45+dS(1))/47.5)^2.5 * (SinD(45-d.lat(1)/2))^4 );   % (A-29)
  nHe  = nHe*dHe;   % (A-30)

  rho(1) = nH*wH + nHe*wHe + nN2*wN2 + nO2*wO2 + nO*wO;   % (A-32)
end;


%--------------------------------------------------------------------------------
% Atmospheric Temperature for Altitudes between 90km and 125km
%   Jacchia Equation A-16
%--------------------------------------------------------------------------------
function tZ = TempLowAlt( z, tX, t90 )

t1  = 1.9*(tX-t90)/35;
dZ  = z - 125;
t4  = 3*( tX - t90 - 2*t1*35/3 )/35^4;
t3  = 4*35*t4/3 - t1/(3*35^2);
tZ  = tX + t1*dZ + t3*dZ.^3 + t4*dZ.^4;   % (A-16)


%--------------------------------------------------------------------------------
% Mean Molecular Mass for Altitudes Less than 105km
%   Jacchia Equation A-18
%--------------------------------------------------------------------------------
function eM = MolMassLowAlt( z )

dZ     = z - 100;
dZSq   = dZ.*dZ;
dZCu   = dZSq.*dZ;
dZ4    = dZCu.*dZ;
dZ5    = dZ4.*dZ;
dZ6    = dZ5.*dZ;
eM   = 28.15204 - 0.085586*dZ + 1.2840e-4*dZSq - 1.0056e-5*dZCu - 1.0210e-5*dZ4 ...
          + 1.5044e-6*dZ5 + 9.9826e-8*dZ6;   % (A-18)


%--------------------------------------------------------------------------------
% Atmospheric Temperature for Altitudes greater than 125km
%   Jacchia Equation A-17
%--------------------------------------------------------------------------------
function tZ = TempHighAlt( z, tX, t90, tE )

t1 = 1.9*(tX-t90)/35;
dZ = z - 125;
a2 = 2*(tE-tX)/pi;
tZ = tX + a2*atan( (t1.*dZ.*(1+(4.5e-6)*dZ.^2.5))./a2 );   % (A-17)
```

```
%---------------------------------------------------------------------------
% Put ASinD(x) in the same quadrant as z
%---------------------------------------------------------------------------
function y = ASinDSameQuadrant( a, z )

y = ASinD(a);

l = find( z > 90 );
if( ~isempty(z) )
  y(l) =  180 - y(l);
end

l = find( z < -90 );
if( ~isempty(z) )
  y(l) = -180 - y(l);
end


%---------------------------------------------------------------------------
% Limit x to the range xMin, xMax
%---------------------------------------------------------------------------
function y = Range( x, xMin, xMax )

md = abs(xMax-xMin);
y  = rem( x, md );
l  = find( y < xMin );
if( ~isempty(l) )
  y(l) = y(l) + md;
end
l  = find( y > xMax );
if( ~isempty(l) )
  y(l) = y(l) - md;
end

%---------------------------------------------------------------------------
% Trigonometric functions
%---------------------------------------------------------------------------
function y = SinD( x )
y = sin( x * pi/180 );

function y = CosD( x )
y = cos( x * pi/180 );

function y = TanD( x )
y = tan( x * pi/180 );

function y = ASinD( x )
y = asin( x )*180/pi;

function y = ATanD( x )
y = atan( x )*180/pi;
```

## A.18 AGravity.m

```
function [aG, aS, aZ, aT] = AGravity( nZ, nT, r, lambda, theta, s, c, j, mu, a )
%-------------------------------------------------------------------------------
%   Compute the gravitational acceleration in spherical
%   coordinates. Acceleration vectors are a [ a(r), a(lambda), a(theta) ].
%   The GEM-T1 coefficients should be unnormalized.
%
%   [s, c, j, mu, a] = LoadGEM( 1 )
%
%   for k = 1:kMax
%     [a, aS, aZ, aT] = AGravity( nZ, nT, r, lambda, theta, s, c, j, mu, a );
%   end
%
%   than
%
%   for k = 1:kMax
%     [a, aS, aZ, aT] = AGravity( nZ, nT, r, lambda, theta );
%   end
%-------------------------------------------------------------------------------
%   Form:
%   [aG, aS, aZ, aT] = AGravity( nZ, nT, r, lambda, theta, s, c, j, mu, a )
%-------------------------------------------------------------------------------
%   ------
%   Inputs
%   ------
%   nZ                    Highest zonal harmonic (m = 0) (empty gives the max #)
%   nT                    Highest sectorial & tesseral harmonic(empty gives max #)
%   r                     Radius
%   lambda                Equatorial angle
%   theta                 Angle from pole
%   s           (36,36)   S terms
%   c           (36,36)   C terms
%   j             (36)    m = 0 terms
%   mu                    Spherical gravitational potential
%   a                     Earth radius
%   -------
%   Outputs
%   -------
%   aG          (3,1)     Total gravitational acceleration m/sec^2
%   aS          (3,1)     Spherical term                   m/sec^2
%   aZ          (3,1)     Zonal term                       m/sec^2
%   aT          (3,1)     Tesseral term                    m/sec^2
%
%-------------------------------------------------------------------------------

%-------------------------------------------------------------------------------
%   Copyright 1996 Princeton Satellite Systems, Inc. All rights reserved.
%-------------------------------------------------------------------------------
if( nargin < 6 )
  [s, c, j, mu, a] = LoadGEM( 1 );
end
```

A-36

```
if( isempty( nZ ) )
  nZ = 36;
end

if( isempty( nT ) )
  nT = 36;
end

% Check the indexes
%-----------------
if ( nZ > 36 ),
  error('Highest zonal harmonic is 36')
end

if ( nT > 36 ),
  error('Highest tesseral harmonic is 36')
end

if ( r == 0 ),
  aG = [ 0 0 0 ]';
  aS = [ 0 0 0 ]';
  aZ = [ 0 0 0 ]';
  aT = [ 0 0 0 ]';
  return
end

% Spherical gravity radial term
%-----------------------------
muORSq = mu/r^2;

% Set up the vectors and compute the spherical earth acceleration vector
%----------------------------------------------------------------------
aS = [ -muORSq; 0; 0 ];
aZ = [ 0; 0; 0 ];
aT = [ 0; 0; 0 ];

% Return if only the spherical earth model is requested
%-----------------------------------------------------
if( nZ == 0 & nT == 0 )
  aG = aS;
  return;
end

% Compute powers of a/r
%---------------------
zTMax = max(nZ,nT);
aOR   = zeros(1,zTMax);
aORK  = zeros(1,zTMax);

aOR(1)  = a/r;
```

```
aORK(1) = 2*aOR(1);
for n = 2:zTMax
  aOR (n) =        aOR(n-1)*aOR(1);
  aORK(n) = (n+1)*aOR(n);
end
aORK = -aORK;

% PDAL returns p(n+1,m+1)
%------------------------
sTheta = sin(theta);
[p, pD] = PDAL( zTMax, nT, cos(theta), -sTheta );
rZ       = 2:(nZ+1);

% Compute the zonal accelerations
%--------------------------------
aZ  = muORSq*[ sum( aORK(1:nZ).*j(1:nZ).* p(rZ,1)');...
               0;...
               sum(  aOR(1:nZ).*j(1:nZ).*pD(rZ,1)')];

% Compute the tesseral and sectorial accelerations
%--------------------------------------------------
rP       = 2:(nT+1);
p        =  p(rP,rP);
pD       = pD(rP,rP);

% sin(m*lambda), cos(m*lambda)
%-----------------------------
if( nT > 0 )
  [sL, cL] = SCHarm( lambda, nT );
end

% Sum over n
%-----------
for n = 1:nT
  m  = 1:n;
  cS = c(n,m).*cL(m) + s(n,m).*sL(m);
  aT = aT + [ aORK(n)*sum(p(n,m).*cS); 0; aOR(n)*sum(pD(n,m).*cS) ];
  if( sTheta ~= 0 )
    cS    = m.*(s(n,m).*cL(m) - c(n,m).*sL(m));
    aT(2) = aT(2) + aOR(n)*sum(p(n,m).*cS)/sTheta;
  end
end

aT = muORSq*aT;

% Total acceleration
%-------------------
aG = aS + aZ + aT;
```

## A.19 JSp2Cart.m

```
function jC = JSp2Cart( s )
%------------------------------------------------------------------------
%   Computes the Jacobian for converting from spherical to
%   cartesian coordinates. Spherical coordinates are defined as
%   [r,theta,phi] where phi is the angle from the +z axis and theta
%   is the angle from +x in the xy-plane.
%
%   x = r*cos(theta)*sin(phi)
%   y = r*sin(theta)*sin(phi)
%   z = r*cos(phi)
%
%------------------------------------------------------------------------
%   Form:
%   jC = JSp2Cart( s )
%------------------------------------------------------------------------
%
%   ------
%   Inputs
%   ------
%   s                 (3,1)  Spherical coordinates [r,theta,phi]
%
%   -------
%   Outputs
%   -------
%   jC                (3,3)  Jacobian from spherical to cartesian
%                             [ x/r x/theta x/phi ]
%                             [ y/r y/theta y/phi ]
%                             [ z/r z/theta z/phi ]
%
%------------------------------------------------------------------------

%------------------------------------------------------------------------
%   Copyright 1993 Princeton Satellite Systems, Inc. All rights reserved.
%------------------------------------------------------------------------

r  = s(1);

cQ = cos(s(2));
sQ = sin(s(2));

cF = cos(s(3));
sF = sin(s(3));

jC = [[cQ*sF;sQ*sF;cF],r*[-sQ*sF,cQ*cF;cQ*sF,sQ*cF;0,-sF]];
```

# Appendix B. Sample Output

## B.1 Model Input File

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%MODEL.M%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  ENVIRONMENTAL DISTURBANCE MODELLING FOR LARGE INFLATABLE STRUCTURES
%  CAPT DONALD J DAVIS, AFIT/ENY, GSO-01M
%
%  NEWMODEL.M -- THIS FILE PROVIDES THE TEMPLATE FOR INPUTTING MODEL DATA INTO
%  FORCES.M FOR ANALYSIS.
%
%  IMPORTANT!!!!--FORCES.M SEARCHES FOR AND LOADS A FILE CALLED MODEL.M.  BEFORE
%  ENTERING A NEW FINITE ELEMENT MODEL, ENSURE PREVIOUS MODELS ARE RENAMED AND
%  BACKED UP.  THEN SAVE THIS FILE AS MODEL.M BEFORE ENTERING ANY DATA INTO THE
%  TEMPLATE


%%%%%%%%%%%%%%%%%%%%%%%%INPUT CLASSICAL ORBITAL ELEMENTS%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  DESIGNATE ORBIT OF INTEREST.  NON-ZERO VALUES MUST BE ENTERED FOR ECCENTRICITY
%  AND INCLINATION. (i.e.  circular orbit-- e=0.0000001)


a=6878.135;                %semi-major axis (km)
e=0.0000000001;            %eccentricity
xi=18.5;                   %inclination (degrees)
argp=245.5774;             %argument of perigee (degrees)
anode=31.5195;             %right ascension of the ascending node
To=[2000,1,12,21,58,23.86];%time of perigee passage[year month day hour minute sec]


Per=2*pi*sqrt((a^3)/3.98601e5);  %orbital period (secs)


%%%%%%%%%%%%%%%%%INPUT SPACE ENVIRONMENTAL DATA%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%    THIS DATA CAN BE FOUND ON THE NOAA SPACE ENVIRONMENT CENTER WEBSITE
%    http://www.sec.noaa.gov/data/geomag.html
%    THESE INDICIES ARE USED IN ATMOSPHERIC DENSITY CALCULATIONS


Ap=12;      %Planetary geomagnetic index 6.7 hours before the calculation
f10=140;    %Daily 10.7cm solar flux (e-22 watts/m^2/cycle/sec)
fhat=181.7; %81-day mean of f (e-22 watts/m^2/cycle/sec)
f400=157.4; %fhat 400 days before computation date


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%SIMULATION RUN PARAMETERS%%%%%%%%%%%%%%%%%%%%%%%%%%

Tint=[2000,1,28,13,25,41]; %Start time for analysis [year month day hour min sec]
freq=4;                    %frequency of calculations (looks per orbit)
dur=1;                     %duration of sim run (# of orbital periods)
picnum=4;                  %# of graphic snapshots during sim run - graphic output
                           %will be displayed at evenly spaced intervals
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%MATERIAL PROPERTIES%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  THE DATA STRUCTURE "MAT" CONTAINS THE MATERIAL PROPERTIES USED IN THE FINITE
%  ELEMENT MODEL.  THE STRUCTURE CAN BE UPDATED AS NECESSARY FOR EACH MODEL BY
%  ADDING NEW ELEMENTS.  ENSURE PROPERTIES ARE ENTERED IN THE APPROPRIATE UNITS.
%  EACH ELEMENT OF THE DATA STRUCTURE HAS SIX FIELDS
%
%  MAT(k).NAME  -TRADE NAME OF THE MATERIAL USED.
%  MAT(k).DENS  -THE DENSITY OF THE MATERIAL (KG/M^3)
%  MAT(k).REFL  -THE COEFFICIENT OF REFLECTION OF THE MATERIAL IN THE SOLAR
%                SPECTRUM(UV-VISIBLE), UNITLESS, 0-1
%  MAT(k).ABSP  -SOLAR ABSORPTIVITY, UNITLESS, 0-1
%  MAT(k).SPEC  -THE SPECULAR REFLECTION COEFFICIENT, THE AMOUNT OF THE
%                REFLECTED LIGHT WHICH IS REFLECTED SPECULARLY, UNITLESS,0-1
%  MAT(k).EMM   -THERMAL EMISSIVITY, UNITLESS, 0-1
%
%  k IS THE MATERIAL INDEX, USED TO ACCESS THE CORRECT ELEMENT OF THE STRUCTURE

mat(1).name='kapton';  %assumed values
mat(1).dens=1420;
mat(1).refl=0.5;
mat(1).absp=0.5;
mat(1).spec=1.0;
mat(1).emm=0.5;

mat(2).name='metallicized mylar';  %assumed values
mat(2).dens=1400;
mat(2).refl=0.9;
mat(2).absp=0.1;
mat(2).spec=0.8;
mat(2).emm=0.6;

mat(3).name='neoprene coated kevlar';
mat(3).dens=1360;
mat(3).refl=0.631;
mat(3).absp=0.369;
mat(3).spec=0.5;  %assumed value
mat(3).emm=0.88;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%STRUCTURE COMPONENTS%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  IN ORDER TO SIMPLIFY DESCRIPTION OF COMPLEX BODIES, THE SATELLITE STRUCTURE
%  CAN BE ANALYZED AS A COLLECTION OF BASIC 3-D SHAPES (i.e. cylinder,sphere,box)
%  FOR EACH COMPONENT, A LOCAL COORDINATE SYSTEM SHOULD BE DEFINED WITH ITS ORGIN
%  INSIDE THE COMPONENT (THE CENTROID IS A CONVENIENT CHOICE.  THE LOCATION OF
%  EACH NODE IN THAT COMPONENT CAN THEN BE DESCRIBED IN TERMS OF THE LOCAL
%  REFERENCE FRAME. ONE COORDINATE SYSTEM IN THE BODY MUST BE DESIGNATED AS THE
%  BASE REFERENCE FRAME.  THE LOCATION AND ORIENTATION OF ALL LOCAL COORDINATE
%  SYSTEMS MUST THEN BE DESCRIBED RELATIVE TO THE BASE FRAME.  THE COMPONENT DATA
%  STRUCTURE BELOW HOLDS THE DATA WHICH DESCRIBES EACH COMPONENT OF THE SATELLITE
```

```
%  SYSTEM, INCLUDING THE LOCATION AND ORIENTATION OF THE LOCAL COORDINATE SYSTEM.
%
%  CAUTION:  THE FIRST ELEMENT OF THE DATA STRUCTURE MUST CONTAIN THE BASE FRAME
%            FOR THE SYSTEM!
%
%  EACH ELEMENT OF THE COMPONENT DATA STRUCTURE HAS 4 FIELDS.
%
%  COMP(i).SHAPE    THIS DESCRIPTOR OF THE COMPONENT SERVES ONLY AS AN AID FOR THE
%                   USER TO AVOID CONFUSION IN MULTI-COMPONENT MODELS (ie LONG BOX,
%                   CYLINDER, SPHERE1, SPHERE2)
%  COMP(i).ORGIN    A 3X1 VECTOR CONTAINING THE CARTESIAN COORDINATES [X;Y;Z] OF
%                   THE LOCAL COORDINATE SYSTEM'S ORGIN RELATIVE TO THE BASE FRAME.
%                   DISTANCES MUST BE ENTERED IN METERS.  SINCE THE FIRST ELEMENT
%                   OF THE STRUCTURE CONTAINS THE BASE FRAME, ITS ORGIN IS ALWAYS
%                   [0;0;0]
%  COMP(i).EULER    THIS IS A 3X1 VECTOR WHICH CONTAINS THE EULER ROTATION ANGLES
%                   [THETA1;THETA2;THETA3] REQUIRED TO TRANSFORM FROM THE BASE
%                   FRAME TO THE LOCAL FRAME USING AN EULER 1-2-3 SEQUENCE. ANGLES
%                   ARE INPUT IN DEGREES.  SINCE THE FIRST ELEMENT CONTAINS THE
%                   BASE FRAME, ITS ROTATION ANGLES WILL ALWAYS BE [0;0;0]
%  COMP(i).TYPE THIS FIELD ALLOWS THE USER TO SELECT WHICH TYPE OF COORDINATES
%                   WILL BE USED TO LOCATE NODES WITHIN EACH COMPONENT.
%
%                        1-CARTESIAN COORDINATES
%                        2-CYLINDRICAL COORDINATES
%                        3-SPHERICAL COORDINATES
%                        4-TOROIDAL COORDINATES
%
%  i IS THE COMPONENT INDEX, USED TO ACCESS THE CORRECT ELEMENT OF THE STRUCTURE

len=2*cos(pi/6);
crad=2*sin(pi/6);

comp(1).shape='sphere1';
comp(1).orgin=[0;0;0];
comp(1).euler=[0;0;0];
comp(1).type=3;

comp(2).shape='cylinder';
comp(2).orgin=[0;0;2+len];
comp(2).euler=[0;0;0];
comp(2).type=2;

comp(3).shape='sphere2';
comp(3).orgin=[0;0;4+2*len];
comp(3).euler=[0;0;0];
comp(3).type=3;
```

%%%%%%%%%%%%%%%%%%%%%%%%%%%ORIENTATION OF THE BODY IN ORBIT%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  THE ORIENTATION OF THE BODY IN ORBIT MUST BE DESCRIBED IN TERMS OF AN EULER
%  1-2-3 ROTATION FROM THE ORBIT FRAME TO THE BODY-FIXED FRAME.  THE BODY-FIXED
%  FRAME IS THE BASE FRAME DEFINED IN THE FIRST ELEMENT OF THE COMP DATA STRUCTURE.
%  THE ORBIT FRAME{a} IS ALIGNED SO THAT THE    a1 AXIS IS ALWAYS ALONG THE S/C
%  RADIUS VECTOR, a3 IS THE ORBIT NORMAL, AND FOR CIRCULAR ORBITS, a2 IS ALIGNED
%  WITH THE VELOCITY VECTOR. ANGLES ARE INPUT IN DEGREES.

phi1=-90;
phi2=90;
phi3=0;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%DEFINE NODE LOCATIONS%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  THE NODE DATA STRUCTURE HOLDS THE LOCATION OF EACH NODE USED IN THE FINITE
%  ELEMENT MODEL. EACH NODE ELEMENT HAS TWO FIELDS
%
%  NODE(j).COORD  THIS FIELD HOLDS THE COORDINATES FOR THE NODE IN TERMS OF THE
%                 LOCAL REFERENCE FRAME.  IT IS EITHER A 3X1 OR 4X1 VECTOR,
%                 DEPENDING ON THE TYPE OF COORDINATE SYSTEM USED.  COORDINATES
%                 ARE ENTERED AS FOLLOWS:
%
%                 CARTESIAN      [X;Y;Z]
%                 CYLINDRICAL    [r;theta;z]
%                 SPHERICAL      [r;theta;phi]
%                 TOROIDAL       [R;theta;r;phi]
%
%     In cylindrical and spherical coordinates, theta is the angle measured counter-
%     clockwise from the positive x axis to the projection of the radius vector into
%     the x-y plane. Phi is the angle between the positive z axis and the radius
%     vector.  In toroidal coordinates, R is the radius of curvature of the torus,
%     and r is the cross-sectional radius.  The coordinate system has its orgin at
%     the center of mass and is aligned so that the z axis is the maximum moment of
%     inertia axis and the x-y plane bisects the torus(see Davis Thesis, Chap 4 or
%     any mechanics textbook for a picture).  Theta is the angle measured counter-
%     clockwise from the positive x axis to R.  Phi is the angle meaured clockwise
%     from the positive z axis to r.
%
%     ALL DISTANCES ARE ENTERED IN METERS AND ANGLES ARE INPUT IN DEGREES!
%
%  NODE(j).COMP   THIS FIELD TELLS THE CODE WHICH COMPONENT OF THE MODEL THE NODE
%                 IS LOCATED ON AND THEREFORE WHICH TYPE OF COORDINATES ARE USED
%                 TO DEFINE IT.  FOR NODES LOCATED AT THE JUNCTION OF TWO
%                 COMPONENTS, EITHER COMPONENT MAY BE SELECTED, WHICHEVER HAS THE
%                 MORE CONVENIENT COORDINATE SYSTEM FOR DESCRIBING THE NODE LOCATION.
%                 THE ENTRY FOR THIS FIELD IS i, WHERE i IS THE COMPONENT INDEX
%                 DEFINED IN THE COMP DATA STRUCTURE
%
%     j IS THE NODE INDEX, USED TO IDENTIFY THE NODE

```
node(1).coord=[2;0;150];
node(1).comp=1;

node(2).coord=[2;30;150];
node(2).comp=1;

node(3).coord=[2;60;150];
node(3).comp=1;

node(4).coord=[2;90;150];
node(4).comp=1;

node(5).coord=[2;120;150];
node(5).comp=1;

node(6).coord=[2;150;150];
node(6).comp=1;

node(7).coord=[2;180;150];
node(7).comp=1;

node(8).coord=[2;210;150];
node(8).comp=1;

node(9).coord=[2;240;150];
node(9).comp=1;

node(10).coord=[2;270;150];
node(10).comp=1;

node(11).coord=[2;300;150];
node(11).comp=1;

node(12).coord=[2;330;150];
node(12).comp=1;

node(13).coord=[2;0;120];
node(13).comp=1;

node(14).coord=[2;30;120];
node(14).comp=1;

node(15).coord=[2;60;120];
node(15).comp=1;

node(16).coord=[2;90;120];
node(16).comp=1;

node(17).coord=[2;120;120];
node(17).comp=1;
```

```
node(18).coord=[2;150;120];
node(18).comp=1;

node(19).coord=[2;180;120];
node(19).comp=1;

node(20).coord=[2;210;120];
node(20).comp=1;

node(21).coord=[2;240;120];
node(21).comp=1;

node(22).coord=[2;270;120];
node(22).comp=1;

node(23).coord=[2;300;120];
node(23).comp=1;

node(24).coord=[2;330;120];
node(24).comp=1;

node(25).coord=[2;0;90];
node(25).comp=1;

node(26).coord=[2;30;90];
node(26).comp=1;

node(27).coord=[2;60;90];
node(27).comp=1;

node(28).coord=[2;90;90];
node(28).comp=1;

node(29).coord=[2;120;90];
node(29).comp=1;

node(30).coord=[2;150;90];
node(30).comp=1;

node(31).coord=[2;180;90];
node(31).comp=1;

node(32).coord=[2;210;90];
node(32).comp=1;

node(33).coord=[2;240;90];
node(33).comp=1;

node(34).coord=[2;270;90];
node(34).comp=1;
```

```
node(35).coord=[2;300;90];
node(35).comp=1;

node(36).coord=[2;330;90];
node(36).comp=1;

node(37).coord=[2;0;60];
node(37).comp=1;

node(38).coord=[2;30;60];
node(38).comp=1;

node(39).coord=[2;60;60];
node(39).comp=1;

node(40).coord=[2;90;60];
node(40).comp=1;

node(41).coord=[2;120;60];
node(41).comp=1;

node(42).coord=[2;150;60];
node(42).comp=1;

node(43).coord=[2;180;60];
node(43).comp=1;

node(44).coord=[2;210;60];
node(44).comp=1;

node(45).coord=[2;240;60];
node(45).comp=1;

node(46).coord=[2;270;60];
node(46).comp=1;

node(47).coord=[2;300;60];
node(47).comp=1;

node(48).coord=[2;330;60];
node(48).comp=1;

node(49).coord=[0;0;0];
node(49).comp=1;

node(50).coord=[2;0;180];
node(50).comp=1;

node(51).coord=[crad;0;-2];
node(51).comp=2;
```

```
node(52).coord=[crad;30;-2];
node(52).comp=2;

node(53).coord=[crad;60;-2];
node(53).comp=2;

node(54).coord=[crad;90;-2];
node(54).comp=2;

node(55).coord=[crad;120;-2];
node(55).comp=2;

node(56).coord=[crad;150;-2];
node(56).comp=2;

node(57).coord=[crad;180;-2];
node(57).comp=2;

node(58).coord=[crad;210;-2];
node(58).comp=2;

node(59).coord=[crad;240;-2];
node(59).comp=2;

node(60).coord=[crad;270;-2];
node(60).comp=2;

node(61).coord=[crad;300;-2];
node(61).comp=2;

node(62).coord=[crad;330;-2];
node(62).comp=2;

node(63).coord=[crad;0;-1];
node(63).comp=2;

node(64).coord=[crad;30;-1];
node(64).comp=2;

node(65).coord=[crad;60;-1];
node(65).comp=2;

node(66).coord=[crad;90;-1];
node(66).comp=2;

node(67).coord=[crad;120;-1];
node(67).comp=2;

node(68).coord=[crad;150;-1];
node(68).comp=2;
```

```
node(69).coord=[crad;180;-1];
node(69).comp=2;

node(70).coord=[crad;210;-1];
node(70).comp=2;

node(71).coord=[crad;240;-1];
node(71).comp=2;

node(72).coord=[crad;270;-1];
node(72).comp=2;

node(73).coord=[crad;300;-1];
node(73).comp=2;

node(74).coord=[crad;330;-1];
node(74).comp=2;

node(75).coord=[crad;0;0];
node(75).comp=2;

node(76).coord=[crad;30;0];
node(76).comp=2;

node(77).coord=[crad;60;0];
node(77).comp=2;

node(78).coord=[crad;90;0];
node(78).comp=2;

node(79).coord=[crad;120;0];
node(79).comp=2;

node(80).coord=[crad;150;0];
node(80).comp=2;

node(81).coord=[crad;180;0];
node(81).comp=2;

node(82).coord=[crad;210;0];
node(82).comp=2;

node(83).coord=[crad;240;0];
node(83).comp=2;

node(84).coord=[crad;270;0];
node(84).comp=2;

node(85).coord=[crad;300;0];
node(85).comp=2;
```

```
node(86).coord=[crad;330;0];
node(86).comp=2;

node(87).coord=[crad;0;1];
node(87).comp=2;

node(88).coord=[crad;30;1];
node(88).comp=2;

node(89).coord=[crad;60;1];
node(89).comp=2;

node(90).coord=[crad;90;1];
node(90).comp=2;

node(91).coord=[crad;120;1];
node(91).comp=2;

node(92).coord=[crad;150;1];
node(92).comp=2;

node(93).coord=[crad;180;1];
node(93).comp=2;

node(94).coord=[crad;210;1];
node(94).comp=2;

node(95).coord=[crad;240;1];
node(95).comp=2;

node(96).coord=[crad;270;1];
node(96).comp=2;

node(97).coord=[crad;300;1];
node(97).comp=2;

node(98).coord=[crad;330;1];
node(98).comp=2;

node(99).coord=[crad;0;2];
node(99).comp=2;

node(100).coord=[crad;30;2];
node(100).comp=2;

node(101).coord=[crad;60;2];
node(101).comp=2;

node(102).coord=[crad;90;2];
node(102).comp=2;
```

```
node(103).coord=[crad;120;2];
node(103).comp=2;

node(104).coord=[crad;150;2];
node(104).comp=2;

node(105).coord=[crad;180;2];
node(105).comp=2;

node(106).coord=[crad;210;2];
node(106).comp=2;

node(107).coord=[crad;240;2];
node(107).comp=2;

node(108).coord=[crad;270;2];
node(108).comp=2;

node(109).coord=[crad;300;2];
node(109).comp=2;

node(110).coord=[crad;330;2];
node(110).comp=2;

node(111).coord=[2;0;120];
node(111).comp=3;

node(112).coord=[2;30;120];
node(112).comp=3;

node(113).coord=[2;60;120];
node(113).comp=3;

node(114).coord=[2;90;120];
node(114).comp=3;

node(115).coord=[2;120;120];
node(115).comp=3;

node(116).coord=[2;150;120];
node(116).comp=3;

node(117).coord=[2;180;120];
node(117).comp=3;

node(118).coord=[2;210;120];
node(118).comp=3;

node(119).coord=[2;240;120];
node(119).comp=3;
```

```
node(120).coord=[2;270;120];
node(120).comp=3;

node(121).coord=[2;300;120];
node(121).comp=3;

node(122).coord=[2;330;120];
node(122).comp=3;

node(123).coord=[2;0;90];
node(123).comp=3;

node(124).coord=[2;30;90];
node(124).comp=3;

node(125).coord=[2;60;90];
node(125).comp=3;

node(126).coord=[2;90;90];
node(126).comp=3;

node(127).coord=[2;120;90];
node(127).comp=3;

node(128).coord=[2;150;90];
node(128).comp=3;

node(129).coord=[2;180;90];
node(129).comp=3;

node(130).coord=[2;210;90];
node(130).comp=3;

node(131).coord=[2;240;90];
node(131).comp=3;

node(132).coord=[2;270;90];
node(132).comp=3;

node(133).coord=[2;300;90];
node(133).comp=3;

node(134).coord=[2;330;90];
node(134).comp=3;

node(135).coord=[2;0;60];
node(135).comp=3;

node(136).coord=[2;30;60];
node(136).comp=3;
```

```
node(137).coord=[2;60;60];
node(137).comp=3;

node(138).coord=[2;90;60];
node(138).comp=3;

node(139).coord=[2;120;60];
node(139).comp=3;

node(140).coord=[2;150;60];
node(140).comp=3;

node(141).coord=[2;180;60];
node(141).comp=3;

node(142).coord=[2;210;60];
node(142).comp=3;

node(143).coord=[2;240;60];
node(143).comp=3;

node(144).coord=[2;270;60];
node(144).comp=3;

node(145).coord=[2;300;60];
node(145).comp=3;

node(146).coord=[2;330;60];
node(146).comp=3;

node(147).coord=[2;0;30];
node(147).comp=3;

node(148).coord=[2;30;30];
node(148).comp=3;

node(149).coord=[2;60;30];
node(149).comp=3;

node(150).coord=[2;90;30];
node(150).comp=3;

node(151).coord=[2;120;30];
node(151).comp=3;

node(152).coord=[2;150;30];
node(152).comp=3;

node(153).coord=[2;180;30];
node(153).comp=3;
```

```
node(154).coord=[2;210;30];
node(154).comp=3;

node(155).coord=[2;240;30];
node(155).comp=3;

node(156).coord=[2;270;30];
node(156).comp=3;

node(157).coord=[2;300;30];
node(157).comp=3;

node(158).coord=[2;330;30];
node(158).comp=3;

node(159).coord=[2;0;0];
node(159).comp=3;

node(160).coord=[0;0;0];
node(160).comp=3;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%DEFINE ELEMENTS%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  THE INELEMENT DATA STRUCTURE DEFINES EACH FINITE ELEMENT IN THE MODEL. THE CODE
%  CAN ACCEPT POINT MASSES (1 node), LINE MASSES (2 nodes), TRIANGULAR ELEMENTS
%  (3 nodes), AND QUADRILATERAL ELEMENTS (4 nodes).  FOR QUADRILATERALS, AT LEAST
%  ONE PAIR OF OPPOSING SIDES MUST BE PARALLEL.  THE INELEMENT DATA STRUCTURE HAS
%  4 FIELDS:
%
%  INELEMENT(m).NODES   THIS FIELD HOLDS THE INDEX NUMBERS OF THE NODES BOUNDING
%                       THE ELEMENT (ie [5,6,7,8]). IT IS A ROW VECTOR WITH 1,2,3
%                       OR 4 ELEMENTS DEPENDING ON WHAT TYPE OF ELEMENT IS BEING
%                       DEFINED.  ANY OF THE NODES MAY BE ENTERED AS THE FIRST
%                       NODE IN THIS FIELD.  THE REMAINING NODES MUST THEN BE
%                       DESCRIBED IN ORDER GOING EITHER CLOCKWISE OR COUNTER-
%                       CLOCKWISE AROUND THE ELEMENT.  FOR TRIANGULAR ELEMENTS,
%                       EITHER DIRECTION IS ACCEPTABLE.  FOR QUADRILATERAL ELEMENTS
%                       ENSURE THE ORDERING OF THE NODES IS SUCH THAT THE LINE
%                       BETWEEN THE FIRST AND SECOND NODES AND THE LINE BETWEEN THE
%                       THIRD AND FOURTH NODES FORM PARALLEL OPPOSING SIDES.
%  INELEMENT(m).MAT     ENTER THE MATERIAL INDEX, k, CORRESPONDING TO THE MATERIAL
%                       OUT OF WHICH THE ELEMENT IS COMPOSED
%  INELEMENT(m).THICK   ENTER THE THICKNESS OF THE ELEMENT IN METERS.  FOR POINT
%                       AND LINE MASSES, ENTER THE TOTAL MASS OF THE ELEMENT (KG)
%                       IN THIS FIELD
%  INELEMENT(m).COMP    ENTER THE INDEX NUMBER, i, CORRESPONDING TO WHICH COMPONENT
%                       THE ELEMENT IS LOCATED ON.
%
%  m IS THE ELEMENT INDEX NUMBER, USED TO IDENTIFY THE ELEMENT
```

```
inelement(1).nodes=[49];
inelement(1).mat=1;
inelement(1).thick=20;
inelement(1).comp=1;

inelement(2).nodes=[50,1,2];
inelement(2).mat=2;
inelement(2).thick=0.000075;
inelement(2).comp=1;

inelement(3).nodes=[50,2,3];
inelement(3).mat=2;
inelement(3).thick=0.000075;
inelement(3).comp=1;

inelement(4).nodes=[50,3,4];
inelement(4).mat=2;
inelement(4).thick=0.000075;
inelement(4).comp=1;

inelement(5).nodes=[50,4,5];
inelement(5).mat=2;
inelement(5).thick=0.000075;
inelement(5).comp=1;

inelement(6).nodes=[50,5,6];
inelement(6).mat=2;
inelement(6).thick=0.000075;
inelement(6).comp=1;

inelement(7).nodes=[50,6,7];
inelement(7).mat=2;
inelement(7).thick=0.000075;
inelement(7).comp=1;

inelement(8).nodes=[50,7,8];
inelement(8).mat=2;
inelement(8).thick=0.000075;
inelement(8).comp=1;

inelement(9).nodes=[50,8,9];
inelement(9).mat=2;
inelement(9).thick=0.000075;
inelement(9).comp=1;

inelement(10).nodes=[50,9,10];
inelement(10).mat=2;
inelement(10).thick=0.000075;
inelement(10).comp=1;
```

```
inelement(11).nodes=[50,10,11];
inelement(11).mat=2;
inelement(11).thick=0.000075;
inelement(11).comp=1;

inelement(12).nodes=[50,11,12];
inelement(12).mat=2;
inelement(12).thick=0.000075;
inelement(12).comp=1;

inelement(13).nodes=[50,12,1];
inelement(13).mat=2;
inelement(13).thick=0.000075;
inelement(13).comp=1;

inelement(14).nodes=[1,2,14,13];
inelement(14).mat=2;
inelement(14).thick=0.000075;
inelement(14).comp=1;

inelement(15).nodes=[2,3,15,14];
inelement(15).mat=2;
inelement(15).thick=0.000075;
inelement(15).comp=1;

inelement(16).nodes=[3,4,16,15];
inelement(16).mat=2;
inelement(16).thick=0.000075;
inelement(16).comp=1;

inelement(17).nodes=[4,5,17,16];
inelement(17).mat=2;
inelement(17).thick=0.000075;
inelement(17).comp=1;

inelement(18).nodes=[5,6,18,17];
inelement(18).mat=2;
inelement(18).thick=0.000075;
inelement(18).comp=1;

inelement(19).nodes=[6,7,19,18];
inelement(19).mat=2;
inelement(19).thick=0.000075;
inelement(19).comp=1;

inelement(20).nodes=[7,8,20,19];
inelement(20).mat=2;
inelement(20).thick=0.000075;
inelement(20).comp=1;
```

```
inelement(21).nodes=[8,9,21,20];
inelement(21).mat=2;
inelement(21).thick=0.000075;
inelement(21).comp=1;

inelement(22).nodes=[9,10,22,21];
inelement(22).mat=2;
inelement(22).thick=0.000075;
inelement(22).comp=1;

inelement(23).nodes=[10,11,23,22];
inelement(23).mat=2;
inelement(23).thick=0.000075;
inelement(23).comp=1;

inelement(24).nodes=[11,12,24,23];
inelement(24).mat=2;
inelement(24).thick=0.000075;
inelement(24).comp=1;

inelement(25).nodes=[12,1,13,24];
inelement(25).mat=2;
inelement(25).thick=0.000075;
inelement(25).comp=1;

inelement(26).nodes=[13,14,26,25];
inelement(26).mat=2;
inelement(26).thick=0.000075;
inelement(26).comp=1;

inelement(27).nodes=[14,15,27,26];
inelement(27).mat=2;
inelement(27).thick=0.000075;
inelement(27).comp=1;

inelement(28).nodes=[15,16,28,27];
inelement(28).mat=2;
inelement(28).thick=0.000075;
inelement(28).comp=1;

inelement(29).nodes=[16,17,29,28];
inelement(29).mat=2;
inelement(29).thick=0.000075;
inelement(29).comp=1;

inelement(30).nodes=[17,18,30,29];
inelement(30).mat=2;
inelement(30).thick=0.000075;
inelement(30).comp=1;
```

```
inelement(31).nodes=[18,19,31,30];
inelement(31).mat=2;
inelement(31).thick=0.000075;
inelement(31).comp=1;

inelement(32).nodes=[19,20,32,31];
inelement(32).mat=2;
inelement(32).thick=0.000075;
inelement(32).comp=1;

inelement(33).nodes=[20,21,33,32];
inelement(33).mat=2;
inelement(33).thick=0.000075;
inelement(33).comp=1;

inelement(34).nodes=[21,22,34,33];
inelement(34).mat=2;
inelement(34).thick=0.000075;
inelement(34).comp=1;

inelement(35).nodes=[22,23,35,34];
inelement(35).mat=2;
inelement(35).thick=0.000075;
inelement(35).comp=1;

inelement(36).nodes=[23,24,36,35];
inelement(36).mat=2;
inelement(36).thick=0.000075;
inelement(36).comp=1;

inelement(37).nodes=[24,13,25,36];
inelement(37).mat=2;
inelement(37).thick=0.000075;
inelement(37).comp=1;

inelement(38).nodes=[25,26,38,37];
inelement(38).mat=2;
inelement(38).thick=0.000075;
inelement(38).comp=1;

inelement(39).nodes=[26,27,39,38];
inelement(39).mat=2;
inelement(39).thick=0.000075;
inelement(39).comp=1;

inelement(40).nodes=[27,28,40,39];
inelement(40).mat=2;
inelement(40).thick=0.000075;
inelement(40).comp=1;
```

```
inelement(41).nodes=[28,29,41,40];
inelement(41).mat=2;
inelement(41).thick=0.000075;
inelement(41).comp=1;

inelement(42).nodes=[29,30,42,41];
inelement(42).mat=2;
inelement(42).thick=0.000075;
inelement(42).comp=1;

inelement(43).nodes=[30,31,43,42];
inelement(43).mat=2;
inelement(43).thick=0.000075;
inelement(43).comp=1;

inelement(44).nodes=[31,32,44,43];
inelement(44).mat=2;
inelement(44).thick=0.000075;
inelement(44).comp=1;

inelement(45).nodes=[32,33,45,44];
inelement(45).mat=2;
inelement(45).thick=0.000075;
inelement(45).comp=1;

inelement(46).nodes=[33,34,46,45];
inelement(46).mat=2;
inelement(46).thick=0.000075;
inelement(46).comp=1;

inelement(47).nodes=[34,35,47,46];
inelement(47).mat=2;
inelement(47).thick=0.000075;
inelement(47).comp=1;

inelement(48).nodes=[35,36,48,47];
inelement(48).mat=2;
inelement(48).thick=0.000075;
inelement(48).comp=1;

inelement(49).nodes=[36,25,37,48];
inelement(49).mat=2;
inelement(49).thick=0.000075;
inelement(49).comp=1;

inelement(50).nodes=[37,38,52,51];
inelement(50).mat=2;
inelement(50).thick=0.000075;
inelement(50).comp=1;
```

```
inelement(51).nodes=[38,39,53,52];
inelement(51).mat=2;
inelement(51).thick=0.000075;
inelement(51).comp=1;

inelement(52).nodes=[39,40,54,53];
inelement(52).mat=2;
inelement(52).thick=0.000075;
inelement(52).comp=1;

inelement(53).nodes=[40,41,55,54];
inelement(53).mat=2;
inelement(53).thick=0.000075;
inelement(53).comp=1;

inelement(54).nodes=[41,42,56,55];
inelement(54).mat=2;
inelement(54).thick=0.000075;
inelement(54).comp=1;

inelement(55).nodes=[42,43,57,56];
inelement(55).mat=2;
inelement(55).thick=0.000075;
inelement(55).comp=1;

inelement(56).nodes=[43,44,58,57];
inelement(56).mat=2;
inelement(56).thick=0.000075;
inelement(56).comp=1;

inelement(57).nodes=[44,45,59,58];
inelement(57).mat=2;
inelement(57).thick=0.000075;
inelement(57).comp=1;

inelement(58).nodes=[45,46,60,59];
inelement(58).mat=2;
inelement(58).thick=0.000075;
inelement(58).comp=1;

inelement(59).nodes=[46,47,61,60];
inelement(59).mat=2;
inelement(59).thick=0.000075;
inelement(59).comp=1;

inelement(60).nodes=[47,48,62,61];
inelement(60).mat=2;
inelement(60).thick=0.000075;
inelement(60).comp=1;
```

```
inelement(61).nodes=[48,37,51,62];
inelement(61).mat=2;
inelement(61).thick=0.000075;
inelement(61).comp=1;

inelement(62).nodes=[51,52,64,63];
inelement(62).mat=3;
inelement(62).thick=0.002;
inelement(62).comp=2;

inelement(63).nodes=[52,53,65,64];
inelement(63).mat=3;
inelement(63).thick=0.002;
inelement(63).comp=2;

inelement(64).nodes=[53,54,66,65];
inelement(64).mat=3;
inelement(64).thick=0.002;
inelement(64).comp=2;

inelement(65).nodes=[54,55,67,66];
inelement(65).mat=3;
inelement(65).thick=0.002;
inelement(65).comp=2;

inelement(66).nodes=[55,56,68,67];
inelement(66).mat=3;
inelement(66).thick=0.002;
inelement(66).comp=2;

inelement(67).nodes=[56,57,69,68];
inelement(67).mat=3;
inelement(67).thick=0.002;
inelement(67).comp=2;

inelement(68).nodes=[57,58,70,69];
inelement(68).mat=3;
inelement(68).thick=0.002;
inelement(68).comp=2;

inelement(69).nodes=[58,59,71,70];
inelement(69).mat=3;
inelement(69).thick=0.002;
inelement(69).comp=2;

inelement(70).nodes=[59,60,72,71];
inelement(70).mat=3;
inelement(70).thick=0.002;
inelement(70).comp=2;
```

```
inelement(71).nodes=[60,61,73,72];
inelement(71).mat=3;
inelement(71).thick=0.002;
inelement(71).comp=2;

inelement(72).nodes=[61,62,74,73];
inelement(72).mat=3;
inelement(72).thick=0.002;
inelement(72).comp=2;

inelement(73).nodes=[62,51,63,74];
inelement(73).mat=3;
inelement(73).thick=0.002;
inelement(73).comp=2;

inelement(74).nodes=[63,64,76,75];
inelement(74).mat=3;
inelement(74).thick=0.002;
inelement(74).comp=2;

inelement(75).nodes=[64,65,77,76];
inelement(75).mat=3;
inelement(75).thick=0.002;
inelement(75).comp=2;

inelement(76).nodes=[65,66,78,77];
inelement(76).mat=3;
inelement(76).thick=0.002;
inelement(76).comp=2;

inelement(77).nodes=[66,67,79,78];
inelement(77).mat=3;
inelement(77).thick=0.002;
inelement(77).comp=2;

inelement(78).nodes=[67,68,80,79];
inelement(78).mat=3;
inelement(78).thick=0.002;
inelement(78).comp=2;

inelement(79).nodes=[68,69,81,80];
inelement(79).mat=3;
inelement(79).thick=0.002;
inelement(79).comp=2;

inelement(80).nodes=[69,70,82,81];
inelement(80).mat=3;
inelement(80).thick=0.002;
inelement(80).comp=2;
```

```
inelement(81).nodes=[70,71,83,82];
inelement(81).mat=3;
inelement(81).thick=0.002;
inelement(81).comp=2;

inelement(82).nodes=[71,72,84,83];
inelement(82).mat=3;
inelement(82).thick=0.002;
inelement(82).comp=2;

inelement(83).nodes=[72,73,85,84];
inelement(83).mat=3;
inelement(83).thick=0.002;
inelement(83).comp=2;

inelement(84).nodes=[73,74,86,85];
inelement(84).mat=3;
inelement(84).thick=0.002;
inelement(84).comp=2;

inelement(85).nodes=[74,63,75,86];
inelement(85).mat=3;
inelement(85).thick=0.002;
inelement(85).comp=2;

inelement(86).nodes=[75,76,88,87];
inelement(86).mat=3;
inelement(86).thick=0.002;
inelement(86).comp=2;

inelement(87).nodes=[76,77,89,88];
inelement(87).mat=3;
inelement(87).thick=0.002;
inelement(87).comp=2;

inelement(88).nodes=[77,78,90,89];
inelement(88).mat=3;
inelement(88).thick=0.002;
inelement(88).comp=2;

inelement(89).nodes=[78,79,91,90];
inelement(89).mat=3;
inelement(89).thick=0.002;
inelement(89).comp=2;

inelement(90).nodes=[79,80,92,91];
inelement(90).mat=3;
inelement(90).thick=0.002;
inelement(90).comp=2;
```

```
inelement(91).nodes=[80,81,93,92];
inelement(91).mat=3;
inelement(91).thick=0.002;
inelement(91).comp=2;

inelement(92).nodes=[81,82,94,93];
inelement(92).mat=3;
inelement(92).thick=0.002;
inelement(92).comp=2;

inelement(93).nodes=[82,83,95,94];
inelement(93).mat=3;
inelement(93).thick=0.002;
inelement(93).comp=2;

inelement(94).nodes=[83,84,96,95];
inelement(94).mat=3;
inelement(94).thick=0.002;
inelement(94).comp=2;

inelement(95).nodes=[84,85,97,96];
inelement(95).mat=3;
inelement(95).thick=0.002;
inelement(95).comp=2;

inelement(96).nodes=[85,86,98,97];
inelement(96).mat=3;
inelement(96).thick=0.002;
inelement(96).comp=2;

inelement(97).nodes=[86,75,87,98];
inelement(97).mat=3;
inelement(97).thick=0.002;
inelement(97).comp=2;

inelement(98).nodes=[87,88,100,99];
inelement(98).mat=3;
inelement(98).thick=0.002;
inelement(98).comp=2;

inelement(99).nodes=[88,89,101,100];
inelement(99).mat=3;
inelement(99).thick=0.002;
inelement(99).comp=2;

inelement(100).nodes=[89,90,102,101];
inelement(100).mat=3;
inelement(100).thick=0.002;
inelement(100).comp=2;
```

```
inelement(101).nodes=[90,91,103,102];
inelement(101).mat=3;
inelement(101).thick=0.002;
inelement(101).comp=2;

inelement(102).nodes=[91,92,104,103];
inelement(102).mat=3;
inelement(102).thick=0.002;
inelement(102).comp=2;

inelement(103).nodes=[92,93,105,104];
inelement(103).mat=3;
inelement(103).thick=0.002;
inelement(103).comp=2;

inelement(104).nodes=[93,94,106,105];
inelement(104).mat=3;
inelement(104).thick=0.002;
inelement(104).comp=2;

inelement(105).nodes=[94,95,107,106];
inelement(105).mat=3;
inelement(105).thick=0.002;
inelement(105).comp=2;

inelement(106).nodes=[95,96,108,107];
inelement(106).mat=3;
inelement(106).thick=0.002;
inelement(106).comp=2;

inelement(107).nodes=[96,97,109,108];
inelement(107).mat=3;
inelement(107).thick=0.002;
inelement(107).comp=2;

inelement(108).nodes=[97,98,110,109];
inelement(108).mat=3;
inelement(108).thick=0.002;
inelement(108).comp=2;

inelement(109).nodes=[98,87,99,110];
inelement(109).mat=3;
inelement(109).thick=0.002;
inelement(109).comp=2;

inelement(110).nodes=[99,100,112,111];
inelement(110).mat=2;
inelement(110).thick=0.000075;
inelement(110).comp=3;
```

```
inelement(111).nodes=[100,101,113,112];
inelement(111).mat=2;
inelement(111).thick=0.000075;
inelement(111).comp=3;

inelement(112).nodes=[101,102,114,113];
inelement(112).mat=2;
inelement(112).thick=0.000075;
inelement(112).comp=3;

inelement(113).nodes=[102,103,115,114];
inelement(113).mat=2;
inelement(113).thick=0.000075;
inelement(113).comp=3;

inelement(114).nodes=[103,104,116,115];
inelement(114).mat=2;
inelement(114).thick=0.000075;
inelement(114).comp=3;

inelement(115).nodes=[104,105,117,116];
inelement(115).mat=2;
inelement(115).thick=0.000075;
inelement(115).comp=3;

inelement(116).nodes=[105,106,118,117];
inelement(116).mat=2;
inelement(116).thick=0.000075;
inelement(116).comp=3;

inelement(117).nodes=[106,107,119,118];
inelement(117).mat=2;
inelement(117).thick=0.000075;
inelement(117).comp=3;

inelement(118).nodes=[107,108,120,119];
inelement(118).mat=2;
inelement(118).thick=0.000075;
inelement(118).comp=3;

inelement(119).nodes=[108,109,121,120];
inelement(119).mat=2;
inelement(119).thick=0.000075;
inelement(119).comp=3;

inelement(120).nodes=[109,110,122,121];
inelement(120).mat=2;
inelement(120).thick=0.000075;
inelement(120).comp=3;
```

```
inelement(121).nodes=[110,99,111,122];
inelement(121).mat=2;
inelement(121).thick=0.000075;
inelement(121).comp=3;

inelement(122).nodes=[111,112,124,123];
inelement(122).mat=2;
inelement(122).thick=0.000075;
inelement(122).comp=3;

inelement(123).nodes=[112,113,125,124];
inelement(123).mat=2;
inelement(123).thick=0.000075;
inelement(123).comp=3;

inelement(124).nodes=[113,114,126,125];
inelement(124).mat=2;
inelement(124).thick=0.000075;
inelement(124).comp=3;

inelement(125).nodes=[114,115,127,126];
inelement(125).mat=2;
inelement(125).thick=0.000075;
inelement(125).comp=3;

inelement(126).nodes=[115,116,128,127];
inelement(126).mat=2;
inelement(126).thick=0.000075;
inelement(126).comp=3;

inelement(127).nodes=[116,117,129,128];
inelement(127).mat=2;
inelement(127).thick=0.000075;
inelement(127).comp=3;

inelement(128).nodes=[117,118,130,129];
inelement(128).mat=2;
inelement(128).thick=0.000075;
inelement(128).comp=3;

inelement(129).nodes=[118,119,131,130];
inelement(129).mat=2;
inelement(129).thick=0.000075;
inelement(129).comp=3;

inelement(130).nodes=[119,120,132,131];
inelement(130).mat=2;
inelement(130).thick=0.000075;
inelement(130).comp=3;
```

```
inelement(131).nodes=[120,121,133,132];
inelement(131).mat=2;
inelement(131).thick=0.000075;
inelement(131).comp=3;

inelement(132).nodes=[121,122,134,133];
inelement(132).mat=2;
inelement(132).thick=0.000075;
inelement(132).comp=3;

inelement(133).nodes=[122,111,123,134];
inelement(133).mat=2;
inelement(133).thick=0.000075;
inelement(133).comp=3;

inelement(134).nodes=[123,124,136,135];
inelement(134).mat=2;
inelement(134).thick=0.000075;
inelement(134).comp=3;

inelement(135).nodes=[124,125,137,136];
inelement(135).mat=2;
inelement(135).thick=0.000075;
inelement(135).comp=3;

inelement(136).nodes=[125,126,138,137];
inelement(136).mat=2;
inelement(136).thick=0.000075;
inelement(136).comp=3;

inelement(137).nodes=[126,127,139,138];
inelement(137).mat=2;
inelement(137).thick=0.000075;
inelement(137).comp=3;

inelement(138).nodes=[127,128,140,139];
inelement(138).mat=2;
inelement(138).thick=0.000075;
inelement(138).comp=3;

inelement(139).nodes=[128,129,141,140];
inelement(139).mat=2;
inelement(139).thick=0.000075;
inelement(139).comp=3;

inelement(140).nodes=[129,130,142,141];
inelement(140).mat=2;
inelement(140).thick=0.000075;
inelement(140).comp=3;
```

```
inelement(141).nodes=[130,131,143,142];
inelement(141).mat=2;
inelement(141).thick=0.000075;
inelement(141).comp=3;

inelement(142).nodes=[131,132,144,143];
inelement(142).mat=2;
inelement(142).thick=0.000075;
inelement(142).comp=3;

inelement(143).nodes=[132,133,145,144];
inelement(143).mat=2;
inelement(143).thick=0.000075;
inelement(143).comp=3;

inelement(144).nodes=[133,134,146,145];
inelement(144).mat=2;
inelement(144).thick=0.000075;
inelement(144).comp=3;

inelement(145).nodes=[134,123,135,146];
inelement(145).mat=2;
inelement(145).thick=0.000075;
inelement(145).comp=3;

inelement(146).nodes=[135,136,148,147];
inelement(146).mat=2;
inelement(146).thick=0.000075;
inelement(146).comp=3;

inelement(147).nodes=[136,137,149,148];
inelement(147).mat=2;
inelement(147).thick=0.000075;
inelement(147).comp=3;

inelement(148).nodes=[137,138,150,149];
inelement(148).mat=2;
inelement(148).thick=0.000075;
inelement(148).comp=3;

inelement(149).nodes=[138,139,151,150];
inelement(149).mat=2;
inelement(149).thick=0.000075;
inelement(149).comp=3;

inelement(150).nodes=[139,140,152,151];
inelement(150).mat=2;
inelement(150).thick=0.000075;
inelement(150).comp=3;
```

```
inelement(151).nodes=[140,141,153,152];
inelement(151).mat=2;
inelement(151).thick=0.000075;
inelement(151).comp=3;

inelement(152).nodes=[141,142,154,153];
inelement(152).mat=2;
inelement(152).thick=0.000075;
inelement(152).comp=3;

inelement(153).nodes=[142,143,155,154];
inelement(153).mat=2;
inelement(153).thick=0.000075;
inelement(153).comp=3;

inelement(154).nodes=[143,144,156,155];
inelement(154).mat=2;
inelement(154).thick=0.000075;
inelement(154).comp=3;

inelement(155).nodes=[144,145,157,156];
inelement(155).mat=2;
inelement(155).thick=0.000075;
inelement(155).comp=3;

inelement(156).nodes=[145,146,158,157];
inelement(156).mat=2;
inelement(156).thick=0.000075;
inelement(156).comp=3;

inelement(157).nodes=[146,135,147,158];
inelement(157).mat=2;
inelement(157).thick=0.000075;
inelement(157).comp=3;

inelement(158).nodes=[147,148,159];
inelement(158).mat=2;
inelement(158).thick=0.000075;
inelement(158).comp=3;

inelement(159).nodes=[148,149,159];
inelement(159).mat=2;
inelement(159).thick=0.000075;
inelement(159).comp=3;

inelement(160).nodes=[149,150,159];
inelement(160).mat=2;
inelement(160).thick=0.000075;
inelement(160).comp=3;
```

```
inelement(161).nodes=[150,151,159];
inelement(161).mat=2;
inelement(161).thick=0.000075;
inelement(161).comp=3;

inelement(162).nodes=[151,152,159];
inelement(162).mat=2;
inelement(162).thick=0.000075;
inelement(162).comp=3;

inelement(163).nodes=[152,153,159];
inelement(163).mat=2;
inelement(163).thick=0.000075;
inelement(163).comp=3;

inelement(164).nodes=[153,154,159];
inelement(164).mat=2;
inelement(164).thick=0.000075;
inelement(164).comp=3;

inelement(165).nodes=[154,155,159];
inelement(165).mat=2;
inelement(165).thick=0.000075;
inelement(165).comp=3;

inelement(166).nodes=[155,156,159];
inelement(166).mat=2;
inelement(166).thick=0.000075;
inelement(166).comp=3;

inelement(167).nodes=[156,157,159];
inelement(167).mat=2;
inelement(167).thick=0.000075;
inelement(167).comp=3;

inelement(168).nodes=[157,158,159];
inelement(168).mat=2;
inelement(168).thick=0.000075;
inelement(168).comp=3;

inelement(169).nodes=[158,147,159];
inelement(169).mat=2;
inelement(169).thick=0.000075;
inelement(169).comp=3;

inelement(170).nodes=[160];
inelement(170).mat=1;
inelement(170).thick=20;
inelement(170).comp=3;

save model;
```

## B.2  Model Output

| Node# | Coordinates(m) |
|---|---|
| 1 | [ 1.00  0.00 -5.46] |
| 2 | [ 0.87  0.50 -5.46] |
| 3 | [ 0.50  0.87 -5.46] |
| 4 | [ 0.00  1.00 -5.46] |
| 5 | [-0.50  0.87 -5.46] |
| 6 | [-0.87  0.50 -5.46] |
| 7 | [-1.00  0.00 -5.46] |
| 8 | [-0.87 -0.50 -5.46] |
| 9 | [-0.50 -0.87 -5.46] |
| 10 | [-0.00 -1.00 -5.46] |
| 11 | [ 0.50 -0.87 -5.46] |
| 12 | [ 0.87 -0.50 -5.46] |
| 13 | [ 1.73  0.00 -4.73] |
| 14 | [ 1.50  0.87 -4.73] |
| 15 | [ 0.87  1.50 -4.73] |
| 16 | [ 0.00  1.73 -4.73] |
| 17 | [-0.87  1.50 -4.73] |
| 18 | [-1.50  0.87 -4.73] |
| 19 | [-1.73  0.00 -4.73] |
| 20 | [-1.50 -0.87 -4.73] |
| 21 | [-0.87 -1.50 -4.73] |
| 22 | [-0.00 -1.73 -4.73] |
| 23 | [ 0.87 -1.50 -4.73] |
| 24 | [ 1.50 -0.87 -4.73] |
| 25 | [ 2.00  0.00 -3.73] |
| 26 | [ 1.73  1.00 -3.73] |
| 27 | [ 1.00  1.73 -3.73] |
| 28 | [ 0.00  2.00 -3.73] |
| 29 | [-1.00  1.73 -3.73] |
| 30 | [-1.73  1.00 -3.73] |
| 31 | [-2.00  0.00 -3.73] |
| 32 | [-1.73 -1.00 -3.73] |
| 33 | [-1.00 -1.73 -3.73] |
| 34 | [-0.00 -2.00 -3.73] |
| 35 | [ 1.00 -1.73 -3.73] |
| 36 | [ 1.73 -1.00 -3.73] |
| 37 | [ 1.73  0.00 -2.73] |
| 38 | [ 1.50  0.87 -2.73] |
| 39 | [ 0.87  1.50 -2.73] |
| 40 | [ 0.00  1.73 -2.73] |
| 41 | [-0.87  1.50 -2.73] |
| 42 | [-1.50  0.87 -2.73] |

| Node# | Coordinates(m) |
|---|---|
| 43 | [-1.73  0.00 -2.73] |
| 44 | [-1.50 -0.87 -2.73] |
| 45 | [-0.87 -1.50 -2.73] |
| 46 | [-0.00 -1.73 -2.73] |
| 47 | [ 0.87 -1.50 -2.73] |
| 48 | [ 1.50 -0.87 -2.73] |
| 49 | [-0.00  0.00 -3.73] |
| 50 | [ 0.00  0.00 -5.73] |
| 51 | [ 1.00  0.00 -2.00] |
| 52 | [ 0.87  0.50 -2.00] |
| 53 | [ 0.50  0.87 -2.00] |
| 54 | [ 0.00  1.00 -2.00] |
| 55 | [-0.50  0.87 -2.00] |
| 56 | [-0.87  0.50 -2.00] |
| 57 | [-1.00  0.00 -2.00] |
| 58 | [-0.87 -0.50 -2.00] |
| 59 | [-0.50 -0.87 -2.00] |
| 60 | [-0.00 -1.00 -2.00] |
| 61 | [ 0.50 -0.87 -2.00] |
| 62 | [ 0.87 -0.50 -2.00] |
| 63 | [ 1.00  0.00 -1.00] |
| 64 | [ 0.87  0.50 -1.00] |
| 65 | [ 0.50  0.87 -1.00] |
| 66 | [ 0.00  1.00 -1.00] |
| 67 | [-0.50  0.87 -1.00] |
| 68 | [-0.87  0.50 -1.00] |
| 69 | [-1.00  0.00 -1.00] |
| 70 | [-0.87 -0.50 -1.00] |
| 71 | [-0.50 -0.87 -1.00] |
| 72 | [-0.00 -1.00 -1.00] |
| 73 | [ 0.50 -0.87 -1.00] |
| 74 | [ 0.87 -0.50 -1.00] |
| 75 | [ 1.00  0.00  0.00] |
| 76 | [ 0.87  0.50  0.00] |
| 77 | [ 0.50  0.87  0.00] |
| 78 | [ 0.00  1.00  0.00] |
| 79 | [-0.50  0.87  0.00] |
| 80 | [-0.87  0.50  0.00] |
| 81 | [-1.00  0.00  0.00] |
| 82 | [-0.87 -0.50  0.00] |
| 83 | [-0.50 -0.87  0.00] |
| 84 | [-0.00 -1.00  0.00] |
| 85 | [ 0.50 -0.87  0.00] |
| 86 | [ 0.87 -0.50  0.00] |

| Node# | Coordinates(m) | | |
|---|---|---|---|
| 87 | [ 1.00 | 0.00 | 1.00] |
| 88 | [ 0.87 | 0.50 | 1.00] |
| 89 | [ 0.50 | 0.87 | 1.00] |
| 90 | [ 0.00 | 1.00 | 1.00] |
| 91 | [-0.50 | 0.87 | 1.00] |
| 92 | [-0.87 | 0.50 | 1.00] |
| 93 | [-1.00 | 0.00 | 1.00] |
| 94 | [-0.87 | -0.50 | 1.00] |
| 95 | [-0.50 | -0.87 | 1.00] |
| 96 | [-0.00 | -1.00 | 1.00] |
| 97 | [ 0.50 | -0.87 | 1.00] |
| 98 | [ 0.87 | -0.50 | 1.00] |
| 99 | [ 1.00 | 0.00 | 2.00] |
| 100 | [ 0.87 | 0.50 | 2.00] |
| 101 | [ 0.50 | 0.87 | 2.00] |
| 102 | [ 0.00 | 1.00 | 2.00] |
| 103 | [-0.50 | 0.87 | 2.00] |
| 104 | [-0.87 | 0.50 | 2.00] |
| 105 | [-1.00 | 0.00 | 2.00] |
| 106 | [-0.87 | -0.50 | 2.00] |
| 107 | [-0.50 | -0.87 | 2.00] |
| 108 | [-0.00 | -1.00 | 2.00] |
| 109 | [ 0.50 | -0.87 | 2.00] |
| 110 | [ 0.87 | -0.50 | 2.00] |
| 111 | [ 1.73 | 0.00 | 2.73] |
| 112 | [ 1.50 | 0.87 | 2.73] |
| 113 | [ 0.87 | 1.50 | 2.73] |
| 114 | [ 0.00 | 1.73 | 2.73] |
| 115 | [-0.87 | 1.50 | 2.73] |
| 116 | [-1.50 | 0.87 | 2.73] |
| 117 | [-1.73 | 0.00 | 2.73] |
| 118 | [-1.50 | -0.87 | 2.73] |
| 119 | [-0.87 | -1.50 | 2.73] |
| 120 | [-0.00 | -1.73 | 2.73] |
| 121 | [ 0.87 | -1.50 | 2.73] |
| 122 | [ 1.50 | -0.87 | 2.73] |
| 123 | [ 2.00 | 0.00 | 3.73] |
| 124 | [ 1.73 | 1.00 | 3.73] |
| 125 | [ 1.00 | 1.73 | 3.73] |
| 126 | [ 0.00 | 2.00 | 3.73] |
| 127 | [-1.00 | 1.73 | 3.73] |
| 128 | [-1.73 | 1.00 | 3.73] |
| 129 | [-2.00 | 0.00 | 3.73] |
| 130 | [-1.73 | -1.00 | 3.73] |

| Node# | Coordinates(m) |
|---|---|
| 131 | [-1.00 -1.73 3.73] |
| 132 | [-0.00 -2.00 3.73] |
| 133 | [ 1.00 -1.73 3.73] |
| 134 | [ 1.73 -1.00 3.73] |
| 135 | [ 1.73 0.00 4.73] |
| 136 | [ 1.50 0.87 4.73] |
| 137 | [ 0.87 1.50 4.73] |
| 138 | [ 0.00 1.73 4.73] |
| 139 | [-0.87 1.50 4.73] |
| 140 | [-1.50 0.87 4.73] |
| 141 | [-1.73 0.00 4.73] |
| 142 | [-1.50 -0.87 4.73] |
| 143 | [-0.87 -1.50 4.73] |
| 144 | [-0.00 -1.73 4.73] |
| 145 | [ 0.87 -1.50 4.73] |
| 146 | [ 1.50 -0.87 4.73] |
| 147 | [ 1.00 0.00 5.46] |
| 148 | [ 0.87 0.50 5.46] |
| 149 | [ 0.50 0.87 5.46] |
| 150 | [ 0.00 1.00 5.46] |
| 151 | [-0.50 0.87 5.46] |
| 152 | [-0.87 0.50 5.46] |
| 153 | [-1.00 0.00 5.46] |
| 154 | [-0.87 -0.50 5.46] |
| 155 | [-0.50 -0.87 5.46] |
| 156 | [-0.00 -1.00 5.46] |
| 157 | [ 0.50 -0.87 5.46] |
| 158 | [ 0.87 -0.50 5.46] |
| 159 | [-0.00 0.00 5.73] |
| 160 | [-0.00 0.00 3.73] |

| Element# | Comp | Area(m^2) | Mass(kg) | Normal(m) | | | Absp | Emm | Refl | Spec | Centroid(m) | | | Nodes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0.000 | 20.000 | [ 0.0 | 0.0 | 0.0] | 0.500 | 0.500 | 0.500 | 1.000 | [ -0.0 | 0.0 | -3.7] | (49) |
| 2 | 1 | 0.259 | 0.027 | [ 0.3 | 0.1 | -1.0] | 0.100 | 0.600 | 0.900 | 0.800 | [ 0.6 | 0.2 | -5.6] | (50,1,2) |
| 3 | 1 | 0.259 | 0.027 | [ 0.2 | 0.2 | -1.0] | 0.100 | 0.600 | 0.900 | 0.800 | [ 0.5 | 0.5 | -5.6] | (50,2,3) |
| 4 | 1 | 0.259 | 0.027 | [ 0.1 | 0.3 | -1.0] | 0.100 | 0.600 | 0.900 | 0.800 | [ 0.2 | 0.6 | -5.6] | (50,3,4) |
| 5 | 1 | 0.259 | 0.027 | [ -0.1 | 0.3 | -1.0] | 0.100 | 0.600 | 0.900 | 0.800 | [ -0.2 | 0.6 | -5.6] | (50,4,5) |
| 6 | 1 | 0.259 | 0.027 | [ -0.2 | 0.2 | -1.0] | 0.100 | 0.600 | 0.900 | 0.800 | [ -0.5 | 0.5 | -5.6] | (50,5,6) |
| 7 | 1 | 0.259 | 0.027 | [ -0.3 | 0.1 | -1.0] | 0.100 | 0.600 | 0.900 | 0.800 | [ -0.6 | 0.2 | -5.6] | (50,6,7) |
| 8 | 1 | 0.259 | 0.027 | [ -0.3 | -0.1 | -1.0] | 0.100 | 0.600 | 0.900 | 0.800 | [ -0.6 | -0.2 | -5.6] | (50,7,8) |
| 9 | 1 | 0.259 | 0.027 | [ -0.2 | -0.2 | -1.0] | 0.100 | 0.600 | 0.900 | 0.800 | [ -0.5 | -0.5 | -5.6] | (50,8,9) |
| 10 | 1 | 0.259 | 0.027 | [ -0.1 | -0.3 | -1.0] | 0.100 | 0.600 | 0.900 | 0.800 | [ -0.2 | -0.6 | -5.6] | (50,9,10) |
| 11 | 1 | 0.259 | 0.027 | [ 0.1 | -0.3 | -1.0] | 0.100 | 0.600 | 0.900 | 0.800 | [ 0.2 | -0.6 | -5.6] | (50,10,11) |
| 12 | 1 | 0.259 | 0.027 | [ 0.2 | -0.2 | -1.0] | 0.100 | 0.600 | 0.900 | 0.800 | [ 0.5 | -0.5 | -5.6] | (50,11,12) |
| 13 | 1 | 0.259 | 0.027 | [ 0.3 | -0.1 | -1.0] | 0.100 | 0.600 | 0.900 | 0.800 | [ 0.6 | -0.2 | -5.6] | (50,12,1) |
| 14 | 1 | 0.720 | 0.076 | [ 0.7 | 0.2 | -0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [ 1.3 | 0.3 | -5.1] | (1,2,14,13) |
| 15 | 1 | 0.720 | 0.076 | [ 0.5 | 0.5 | -0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [ 1.0 | 1.0 | -5.1] | (2,3,15,14) |
| 16 | 1 | 0.720 | 0.076 | [ 0.2 | 0.7 | -0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [ 0.3 | 1.3 | -5.1] | (3,4,16,15) |
| 17 | 1 | 0.720 | 0.076 | [ -0.2 | 0.7 | -0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [ -0.3 | 1.3 | -5.1] | (4,5,17,16) |
| 18 | 1 | 0.720 | 0.076 | [ -0.5 | 0.5 | -0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [ -1.0 | 1.0 | -5.1] | (5,6,18,17) |
| 19 | 1 | 0.720 | 0.076 | [ -0.7 | 0.2 | -0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [ -1.3 | 0.3 | -5.1] | (6,7,19,18) |
| 20 | 1 | 0.720 | 0.076 | [ -0.7 | -0.2 | -0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [ -1.3 | -0.3 | -5.1] | (7,8,20,19) |
| 21 | 1 | 0.720 | 0.076 | [ -0.5 | -0.5 | -0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [ -1.0 | -1.0 | -5.1] | (8,9,21,20) |
| 22 | 1 | 0.720 | 0.076 | [ -0.2 | -0.7 | -0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [ -0.3 | -1.3 | -5.1] | (9,10,22,21) |
| 23 | 1 | 0.720 | 0.076 | [ 0.2 | -0.7 | -0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [ 0.3 | -1.3 | -5.1] | (10,11,23,22) |
| 24 | 1 | 0.720 | 0.076 | [ 0.5 | -0.5 | -0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [ 1.0 | -1.0 | -5.1] | (11,12,24,23) |
| 25 | 1 | 0.720 | 0.076 | [ 0.7 | -0.2 | -0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [ 1.3 | -0.3 | -5.1] | (12,1,13,24) |
| 26 | 1 | 0.998 | 0.105 | [ 0.9 | 0.3 | -0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ 1.7 | 0.5 | -4.2] | (13,14,26,25) |
| 27 | 1 | 0.998 | 0.105 | [ 0.7 | 0.7 | -0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ 1.3 | 1.3 | -4.2] | (14,15,27,26) |
| 28 | 1 | 0.998 | 0.105 | [ 0.3 | 0.9 | -0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ 0.5 | 1.7 | -4.2] | (15,16,28,27) |
| 29 | 1 | 0.998 | 0.105 | [ -0.3 | 0.9 | -0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ -0.5 | 1.7 | -4.2] | (16,17,29,28) |
| 30 | 1 | 0.998 | 0.105 | [ -0.7 | 0.7 | -0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ -1.3 | 1.3 | -4.2] | (17,18,30,29) |
| 31 | 1 | 0.998 | 0.105 | [ -0.9 | 0.3 | -0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ -1.7 | 0.5 | -4.2] | (18,19,31,30) |
| 32 | 1 | 0.998 | 0.105 | [ -0.9 | -0.3 | -0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ -1.7 | -0.5 | -4.2] | (19,20,32,31) |
| 33 | 1 | 0.998 | 0.105 | [ -0.7 | -0.7 | -0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ -1.3 | -1.3 | -4.2] | (20,21,33,32) |
| 34 | 1 | 0.998 | 0.105 | [ -0.3 | -0.9 | -0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ -0.5 | -1.7 | -4.2] | (21,22,34,33) |

| Element# | Comp | Area(m^2) | Mass(kg) | Normal(m) | | | Absp | Emm | Refl | Spec | Centroid(m) | | | Nodes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 35 | 1 | 0.998 | 0.105 | [ 0.3 | -0.9 | -0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ 0.5 | -1.7 | -4.2] | (22,23,35,34) |
| 36 | 1 | 0.998 | 0.105 | [ 0.7 | -0.7 | -0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ 1.3 | -1.3 | -4.2] | (23,24,36,35) |
| 37 | 1 | 0.998 | 0.105 | [ 0.9 | -0.3 | -0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ 1.7 | -0.5 | -4.2] | (24,13,25,36) |
| 38 | 1 | 0.998 | 0.105 | [ 0.9 | 0.3 | 0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ 1.7 | 0.5 | -3.2] | (25,26,38,37) |
| 39 | 1 | 0.998 | 0.105 | [ 0.7 | 0.7 | 0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ 1.3 | 1.3 | -3.2] | (26,27,39,38) |
| 40 | 1 | 0.998 | 0.105 | [ 0.3 | 0.9 | 0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ 0.5 | 1.7 | -3.2] | (27,28,40,39) |
| 41 | 1 | 0.998 | 0.105 | [-0.3 | 0.9 | 0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [-0.5 | 1.7 | -3.2] | (28,29,41,40) |
| 42 | 1 | 0.998 | 0.105 | [-0.7 | 0.7 | 0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [-1.3 | 1.3 | -3.2] | (29,30,42,41) |
| 43 | 1 | 0.998 | 0.105 | [-0.9 | 0.3 | 0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [-1.7 | 0.5 | -3.2] | (30,31,43,42) |
| 44 | 1 | 0.998 | 0.105 | [-0.9 | -0.3 | 0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [-1.7 | -0.5 | -3.2] | (31,32,44,43) |
| 45 | 1 | 0.998 | 0.105 | [-0.7 | -0.7 | 0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [-1.3 | -1.3 | -3.2] | (32,33,45,44) |
| 46 | 1 | 0.998 | 0.105 | [-0.3 | -0.9 | 0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [-0.5 | -1.7 | -3.2] | (33,34,46,45) |
| 47 | 1 | 0.998 | 0.105 | [ 0.3 | -0.9 | 0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ 0.5 | -1.7 | -3.2] | (34,35,47,46) |
| 48 | 1 | 0.998 | 0.105 | [ 0.7 | -0.7 | 0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ 1.3 | -1.3 | -3.2] | (35,36,48,47) |
| 49 | 1 | 0.998 | 0.105 | [ 0.9 | -0.3 | 0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ 1.7 | -0.5 | -3.2] | (36,25,37,48) |
| 50 | 1 | 0.720 | 0.076 | [ 0.7 | 0.2 | 0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [ 1.3 | 0.3 | -2.4] | (37,38,52,51) |
| 51 | 1 | 0.720 | 0.076 | [ 0.5 | 0.5 | 0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [ 1.0 | 1.0 | -2.4] | (38,39,53,52) |
| 52 | 1 | 0.720 | 0.076 | [ 0.2 | 0.7 | 0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [ 0.3 | 1.3 | -2.4] | (39,40,54,53) |
| 53 | 1 | 0.720 | 0.076 | [-0.2 | 0.7 | 0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [-0.3 | 1.3 | -2.4] | (40,41,55,54) |
| 54 | 1 | 0.720 | 0.076 | [-0.5 | 0.5 | 0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [-1.0 | 1.0 | -2.4] | (41,42,56,55) |
| 55 | 1 | 0.720 | 0.076 | [-0.7 | 0.2 | 0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [-1.3 | 0.3 | -2.4] | (42,43,57,56) |
| 56 | 1 | 0.720 | 0.076 | [-0.7 | -0.2 | 0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [-1.3 | -0.3 | -2.4] | (43,44,58,57) |
| 57 | 1 | 0.720 | 0.076 | [-0.5 | -0.5 | 0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [-1.0 | -1.0 | -2.4] | (44,45,59,58) |
| 58 | 1 | 0.720 | 0.076 | [-0.2 | -0.7 | 0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [-0.3 | -1.3 | -2.4] | (45,46,60,59) |
| 59 | 1 | 0.720 | 0.076 | [ 0.2 | -0.7 | 0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [ 0.3 | -1.3 | -2.4] | (46,47,61,60) |
| 60 | 1 | 0.720 | 0.076 | [ 0.5 | -0.5 | 0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [ 1.0 | -1.0 | -2.4] | (47,48,62,61) |
| 61 | 1 | 0.720 | 0.076 | [ 0.7 | -0.2 | 0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [ 1.3 | -0.3 | -2.4] | (48,37,51,62) |
| 62 | 2 | 0.518 | 1.408 | [ 1.0 | 0.3 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ 0.9 | 0.2 | -1.5] | (51,52,64,63) |
| 63 | 2 | 0.518 | 1.408 | [ 0.7 | 0.7 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ 0.7 | 0.7 | -1.5] | (52,53,65,64) |
| 64 | 2 | 0.518 | 1.408 | [ 0.3 | 1.0 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ 0.3 | 0.9 | -1.5] | (53,54,66,65) |
| 65 | 2 | 0.518 | 1.408 | [-0.3 | 1.0 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [-0.2 | 0.9 | -1.5] | (54,55,67,66) |
| 66 | 2 | 0.518 | 1.408 | [-0.7 | 0.7 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [-0.7 | 0.7 | -1.5] | (55,56,68,67) |
| 67 | 2 | 0.518 | 1.408 | [-1.0 | 0.3 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [-0.9 | 0.3 | -1.5] | (56,57,69,68) |
| 68 | 2 | 0.518 | 1.408 | [-1.0 | -0.3 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [-0.9 | -0.2 | -1.5] | (57,58,70,69) |

| Element# | Comp | Area(m^2) | Mass(kg) | Normal(m) | | | Absp | Emm | Refl | Spec | Centroid(m) | | | Nodes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 69 | 2 | 0.518 | 1.408 | [ -0.7 | -0.7 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ -0.7 | -0.7 | -1.5] | (58,59,71,70) |
| 70 | 2 | 0.518 | 1.408 | [ -0.3 | -1.0 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ -0.3 | -0.9 | -1.5] | (59,60,72,71) |
| 71 | 2 | 0.518 | 1.408 | [ 0.3 | -1.0 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ 0.2 | -0.9 | -1.5] | (60,61,73,72) |
| 72 | 2 | 0.518 | 1.408 | [ 0.7 | -0.7 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ 0.7 | -0.7 | -1.5] | (61,62,74,73) |
| 73 | 2 | 0.518 | 1.408 | [ 1.0 | -0.3 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ 0.9 | -0.3 | -1.5] | (62,51,63,74) |
| 74 | 2 | 0.518 | 1.408 | [ 1.0 | 0.3 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ 0.9 | 0.2 | -0.5] | (63,64,76,75) |
| 75 | 2 | 0.518 | 1.408 | [ 0.7 | 0.7 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ 0.7 | 0.7 | -0.5] | (64,65,77,76) |
| 76 | 2 | 0.518 | 1.408 | [ 0.3 | 1.0 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ 0.3 | 0.9 | -0.5] | (65,66,78,77) |
| 77 | 2 | 0.518 | 1.408 | [ -0.3 | 1.0 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ -0.2 | 0.9 | -0.5] | (66,67,79,78) |
| 78 | 2 | 0.518 | 1.408 | [ -0.7 | 0.7 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ -0.7 | 0.7 | -0.5] | (67,68,80,79) |
| 79 | 2 | 0.518 | 1.408 | [ -1.0 | 0.3 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ -0.9 | 0.3 | -0.5] | (68,69,81,80) |
| 80 | 2 | 0.518 | 1.408 | [ -1.0 | -0.3 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ -0.9 | -0.2 | -0.5] | (69,70,82,81) |
| 81 | 2 | 0.518 | 1.408 | [ -0.7 | -0.7 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ -0.7 | -0.7 | -0.5] | (70,71,83,82) |
| 82 | 2 | 0.518 | 1.408 | [ -0.3 | -1.0 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ -0.3 | -0.9 | -0.5] | (71,72,84,83) |
| 83 | 2 | 0.518 | 1.408 | [ 0.3 | -1.0 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ 0.2 | -0.9 | -0.5] | (72,73,85,84) |
| 84 | 2 | 0.518 | 1.408 | [ 0.7 | -0.7 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ 0.7 | -0.7 | -0.5] | (73,74,86,85) |
| 85 | 2 | 0.518 | 1.408 | [ 1.0 | -0.3 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ 0.9 | -0.3 | -0.5] | (74,63,75,86) |
| 86 | 2 | 0.518 | 1.408 | [ 1.0 | 0.3 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ 0.9 | 0.2 | 0.5] | (75,76,88,87) |
| 87 | 2 | 0.518 | 1.408 | [ 0.7 | 0.7 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ 0.7 | 0.7 | 0.5] | (76,77,89,88) |
| 88 | 2 | 0.518 | 1.408 | [ 0.3 | 1.0 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ 0.3 | 0.9 | 0.5] | (77,78,90,89) |
| 89 | 2 | 0.518 | 1.408 | [ -0.3 | 1.0 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ -0.2 | 0.9 | 0.5] | (78,79,91,90) |
| 90 | 2 | 0.518 | 1.408 | [ -0.7 | 0.7 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ -0.7 | 0.7 | 0.5] | (79,80,92,91) |
| 91 | 2 | 0.518 | 1.408 | [ -1.0 | 0.3 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ -0.9 | 0.3 | 0.5] | (80,81,93,92) |
| 92 | 2 | 0.518 | 1.408 | [ -1.0 | -0.3 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ -0.9 | -0.2 | 0.5] | (81,82,94,93) |
| 93 | 2 | 0.518 | 1.408 | [ -0.7 | -0.7 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ -0.7 | -0.7 | 0.5] | (82,83,95,94) |
| 94 | 2 | 0.518 | 1.408 | [ -0.3 | -1.0 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ -0.3 | -0.9 | 0.5] | (83,84,96,95) |
| 95 | 2 | 0.518 | 1.408 | [ 0.3 | -1.0 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ 0.2 | -0.9 | 0.5] | (84,85,97,96) |
| 96 | 2 | 0.518 | 1.408 | [ 0.7 | -0.7 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ 0.7 | -0.7 | 0.5] | (85,86,98,97) |
| 97 | 2 | 0.518 | 1.408 | [ 1.0 | -0.3 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ 0.9 | -0.3 | 0.5] | (86,75,87,98) |
| 98 | 2 | 0.518 | 1.408 | [ 1.0 | 0.3 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ 0.9 | 0.2 | 1.5] | (87,88,100,99) |
| 99 | 2 | 0.518 | 1.408 | [ 0.7 | 0.7 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ 0.7 | 0.7 | 1.5] | (88,89,101,100) |
| 100 | 2 | 0.518 | 1.408 | [ 0.3 | 1.0 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ 0.3 | 0.9 | 1.5] | (89,90,102,101) |
| 101 | 2 | 0.518 | 1.408 | [ -0.3 | 1.0 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ -0.2 | 0.9 | 1.5] | (90,91,103,102) |
| 102 | 2 | 0.518 | 1.408 | [ -0.7 | 0.7 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ -0.7 | 0.7 | 1.5] | (91,92,104,103) |

| Element# | Comp | Area(m^2) | Mass(kg) | Normal(m) | | | Absp | Emm | Refl | Spec | Centroid(m) | | | Nodes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 103 | 2 | 0.518 | 1.408 | [ -1.0 | 0.3 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ -0.9 | 0.3 | 1.5] | (92,93,105,104) |
| 104 | 2 | 0.518 | 1.408 | [ -1.0 | -0.3 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ -0.9 | -0.2 | 1.5] | (93,94,106,105) |
| 105 | 2 | 0.518 | 1.408 | [ -0.7 | -0.7 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ -0.7 | -0.7 | 1.5] | (94,95,107,106) |
| 106 | 2 | 0.518 | 1.408 | [ -0.3 | -1.0 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ -0.3 | -0.9 | 1.5] | (95,96,108,107) |
| 107 | 2 | 0.518 | 1.408 | [ 0.3 | -1.0 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ 0.2 | -0.9 | 1.5] | (96,97,109,108) |
| 108 | 2 | 0.518 | 1.408 | [ 0.7 | -0.7 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ 0.7 | -0.7 | 1.5] | (97,98,110,109) |
| 109 | 2 | 0.518 | 1.408 | [ 1.0 | -0.3 | 0.0] | 0.369 | 0.880 | 0.631 | 0.500 | [ 0.9 | -0.3 | 1.5] | (98,87,99,110) |
| 110 | 3 | 0.720 | 0.076 | [ 0.7 | 0.2 | -0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [ 1.3 | 0.3 | 2.4] | (99,100,112,111) |
| 111 | 3 | 0.720 | 0.076 | [ 0.5 | 0.5 | -0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [ 1.0 | 1.0 | 2.4] | (100,101,113,112) |
| 112 | 3 | 0.720 | 0.076 | [ 0.2 | 0.7 | -0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [ 0.3 | 1.3 | 2.4] | (101,102,114,113) |
| 113 | 3 | 0.720 | 0.076 | [ -0.2 | 0.7 | -0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [ -0.3 | 1.3 | 2.4] | (102,103,115,114) |
| 114 | 3 | 0.720 | 0.076 | [ -0.5 | 0.5 | -0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [ -1.0 | 1.0 | 2.4] | (103,104,116,115) |
| 115 | 3 | 0.720 | 0.076 | [ -0.7 | 0.2 | -0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [ -1.3 | 0.3 | 2.4] | (104,105,117,116) |
| 116 | 3 | 0.720 | 0.076 | [ -0.7 | -0.2 | -0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [ -1.3 | -0.3 | 2.4] | (105,106,118,117) |
| 117 | 3 | 0.720 | 0.076 | [ -0.5 | -0.5 | -0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [ -1.0 | -1.0 | 2.4] | (106,107,119,118) |
| 118 | 3 | 0.720 | 0.076 | [ -0.2 | -0.7 | -0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [ -0.3 | -1.3 | 2.4] | (107,108,120,119) |
| 119 | 3 | 0.720 | 0.076 | [ 0.2 | -0.7 | -0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [ 0.3 | -1.3 | 2.4] | (108,109,121,120) |
| 120 | 3 | 0.720 | 0.076 | [ 0.5 | -0.5 | -0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [ 1.0 | -1.0 | 2.4] | (109,110,122,121) |
| 121 | 3 | 0.720 | 0.076 | [ 0.7 | -0.2 | -0.7] | 0.100 | 0.600 | 0.900 | 0.800 | [ 1.3 | -0.3 | 2.4] | (110,99,111,122) |
| 122 | 3 | 0.998 | 0.105 | [ 0.9 | 0.3 | -0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ 1.7 | 0.5 | 3.2] | (111,112,124,123) |
| 123 | 3 | 0.998 | 0.105 | [ 0.7 | 0.7 | -0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ 1.3 | 1.3 | 3.2] | (112,113,125,124) |
| 124 | 3 | 0.998 | 0.105 | [ 0.3 | 0.9 | -0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ 0.5 | 1.7 | 3.2] | (113,114,126,125) |
| 125 | 3 | 0.998 | 0.105 | [ -0.3 | 0.9 | -0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ -0.5 | 1.7 | 3.2] | (114,115,127,126) |
| 126 | 3 | 0.998 | 0.105 | [ -0.7 | 0.7 | -0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ -1.3 | 1.3 | 3.2] | (115,116,128,127) |
| 127 | 3 | 0.998 | 0.105 | [ -0.9 | 0.3 | -0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ -1.7 | 0.5 | 3.2] | (116,117,129,128) |
| 128 | 3 | 0.998 | 0.105 | [ -0.9 | -0.3 | -0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ -1.7 | -0.5 | 3.2] | (117,118,130,129) |
| 129 | 3 | 0.998 | 0.105 | [ -0.7 | -0.7 | -0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ -1.3 | -1.3 | 3.2] | (118,119,131,130) |
| 130 | 3 | 0.998 | 0.105 | [ -0.3 | -0.9 | -0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ -0.5 | -1.7 | 3.2] | (119,120,132,131) |
| 131 | 3 | 0.998 | 0.105 | [ 0.3 | -0.9 | -0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ 0.5 | -1.7 | 3.2] | (120,121,133,132) |
| 132 | 3 | 0.998 | 0.105 | [ 0.7 | -0.7 | -0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ 1.3 | -1.3 | 3.2] | (121,122,134,133) |
| 133 | 3 | 0.998 | 0.105 | [ 0.9 | -0.3 | -0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ 1.7 | -0.5 | 3.2] | (122,111,123,134) |
| 134 | 3 | 0.998 | 0.105 | [ 0.9 | 0.3 | 0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ 1.7 | 0.5 | 4.2] | (123,124,136,135) |
| 135 | 3 | 0.998 | 0.105 | [ 0.7 | 0.7 | 0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ 1.3 | 1.3 | 4.2] | (124,125,137,136) |
| 136 | 3 | 0.998 | 0.105 | [ 0.3 | 0.9 | 0.3] | 0.100 | 0.600 | 0.900 | 0.800 | [ 0.5 | 1.7 | 4.2] | (125,126,138,137) |

| Element# | Comp | Area(m^2) | Mass(kg) | Normal(m) | | | Absp | Emm | Refl | Spec | Centroid(m) | | | Nodes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 137 | 3 | 0.998 | 0.105 | [ -0.3 | 0.9 | 0.3 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ -0.5 | 1.7 | 4.2 ] | (126,127,139,138) |
| 138 | 3 | 0.998 | 0.105 | [ -0.7 | 0.7 | 0.3 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ -1.3 | 1.3 | 4.2 ] | (127,128,140,139) |
| 139 | 3 | 0.998 | 0.105 | [ -0.9 | 0.3 | 0.3 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ -1.7 | 0.5 | 4.2 ] | (128,129,141,140) |
| 140 | 3 | 0.998 | 0.105 | [ -0.9 | -0.3 | 0.3 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ -1.7 | -0.5 | 4.2 ] | (129,130,142,141) |
| 141 | 3 | 0.998 | 0.105 | [ -0.7 | -0.7 | 0.3 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ -1.3 | -1.3 | 4.2 ] | (130,131,143,142) |
| 142 | 3 | 0.998 | 0.105 | [ -0.3 | -0.9 | 0.3 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ -0.5 | -1.7 | 4.2 ] | (131,132,144,143) |
| 143 | 3 | 0.998 | 0.105 | [ 0.3 | -0.9 | 0.3 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ 0.5 | -1.7 | 4.2 ] | (132,133,145,144) |
| 144 | 3 | 0.998 | 0.105 | [ 0.7 | -0.7 | 0.3 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ 1.3 | -1.3 | 4.2 ] | (133,134,146,145) |
| 145 | 3 | 0.998 | 0.105 | [ 0.9 | -0.3 | 0.3 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ 1.7 | -0.5 | 4.2 ] | (134,123,135,146) |
| 146 | 3 | 0.720 | 0.076 | [ 0.7 | 0.2 | 0.7 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ 1.3 | 0.3 | 5.1 ] | (135,136,148,147) |
| 147 | 3 | 0.720 | 0.076 | [ 0.5 | 0.5 | 0.7 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ 1.0 | 1.0 | 5.1 ] | (136,137,149,148) |
| 148 | 3 | 0.720 | 0.076 | [ 0.2 | 0.7 | 0.7 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ 0.3 | 1.3 | 5.1 ] | (137,138,150,149) |
| 149 | 3 | 0.720 | 0.076 | [ -0.2 | 0.7 | 0.7 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ -0.3 | 1.3 | 5.1 ] | (138,139,151,150) |
| 150 | 3 | 0.720 | 0.076 | [ -0.5 | 0.5 | 0.7 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ -1.0 | 1.0 | 5.1 ] | (139,140,152,151) |
| 151 | 3 | 0.720 | 0.076 | [ -0.7 | 0.2 | 0.7 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ -1.3 | 0.3 | 5.1 ] | (140,141,153,152) |
| 152 | 3 | 0.720 | 0.076 | [ -0.7 | -0.2 | 0.7 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ -1.3 | -0.3 | 5.1 ] | (141,142,154,153) |
| 153 | 3 | 0.720 | 0.076 | [ -0.5 | -0.5 | 0.7 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ -1.0 | -1.0 | 5.1 ] | (142,143,155,154) |
| 154 | 3 | 0.720 | 0.076 | [ -0.2 | -0.7 | 0.7 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ -0.3 | -1.3 | 5.1 ] | (143,144,156,155) |
| 155 | 3 | 0.720 | 0.076 | [ 0.2 | -0.7 | 0.7 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ 0.3 | -1.3 | 5.1 ] | (144,145,157,156) |
| 156 | 3 | 0.720 | 0.076 | [ 0.5 | -0.5 | 0.7 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ 1.0 | -1.0 | 5.1 ] | (145,146,158,157) |
| 157 | 3 | 0.720 | 0.076 | [ 0.7 | -0.2 | 0.7 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ 1.3 | -0.3 | 5.1 ] | (146,135,147,158) |
| 158 | 3 | 0.259 | 0.027 | [ 0.3 | 0.1 | 1.0 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ 0.6 | 0.2 | 5.6 ] | (147,148,159) |
| 159 | 3 | 0.259 | 0.027 | [ 0.2 | 0.2 | 1:0 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ 0.5 | 0.5 | 5.6 ] | (148,149,159) |
| 160 | 3 | 0.259 | 0.027 | [ 0.1 | 0.3 | 1.0 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ 0.2 | 0.6 | 5.6 ] | (149,150,159) |
| 161 | 3 | 0.259 | 0.027 | [ -0.1 | 0.3 | 1.0 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ -0.2 | 0.6 | 5.6 ] | (150,151,159) |
| 162 | 3 | 0.259 | 0.027 | [ -0.2 | 0.2 | 1.0 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ -0.5 | 0.5 | 5.6 ] | (151,152,159) |
| 163 | 3 | 0.259 | 0.027 | [ -0.3 | 0.1 | 1.0 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ -0.6 | 0.2 | 5.6 ] | (152,153,159) |
| 164 | 3 | 0.259 | 0.027 | [ -0.3 | -0.1 | 1.0 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ -0.6 | -0.2 | 5.6 ] | (153,154,159) |
| 165 | 3 | 0.259 | 0.027 | [ -0.2 | -0.2 | 1.0 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ -0.5 | -0.5 | 5.6 ] | (154,155,159) |
| 166 | 3 | 0.259 | 0.027 | [ -0.1 | -0.3 | 1.0 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ -0.2 | -0.6 | 5.6 ] | (155,156,159) |
| 167 | 3 | 0.259 | 0.027 | [ 0.1 | -0.3 | 1.0 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ 0.2 | -0.6 | 5.6 ] | (156,157,159) |
| 168 | 3 | 0.259 | 0.027 | [ 0.2 | -0.2 | 1.0 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ 0.5 | -0.5 | 5.6 ] | (157,158,159) |
| 169 | 3 | 0.259 | 0.027 | [ 0.3 | -0.1 | 1.0 ] | 0.100 | 0.600 | 0.900 | 0.800 | [ 0.6 | -0.2 | 5.6 ] | (158,147,159) |
| 170 | 3 | 0.000 | 20.000 | [ 0.0 | 0.0 | 0.0 ] | 0.500 | 0.500 | 0.500 | 1.000 | [ -0.0 | 0.0 | 3.7 ] | (160) |
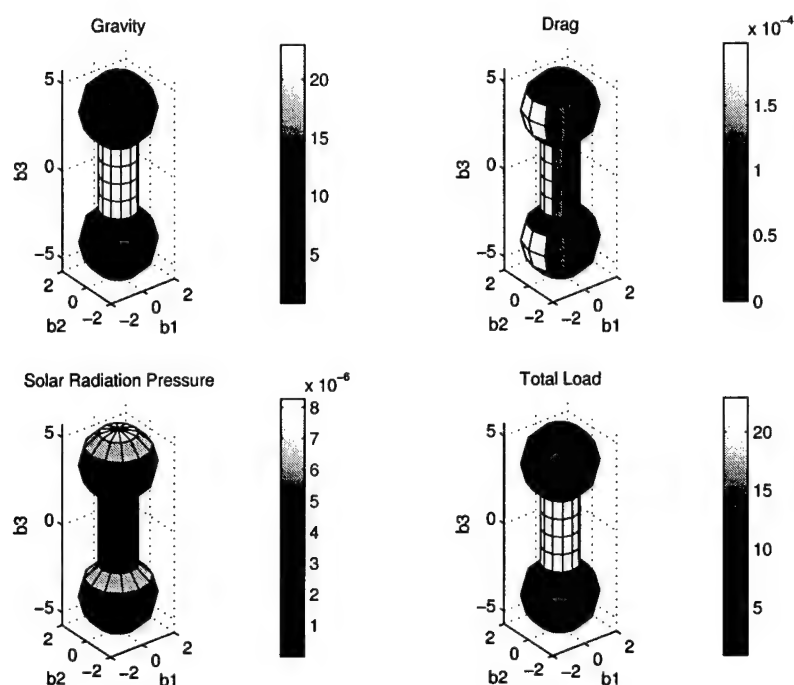
Figure B.1    Mechanical Loads (Pa), h=500km, $\eta = 0°$
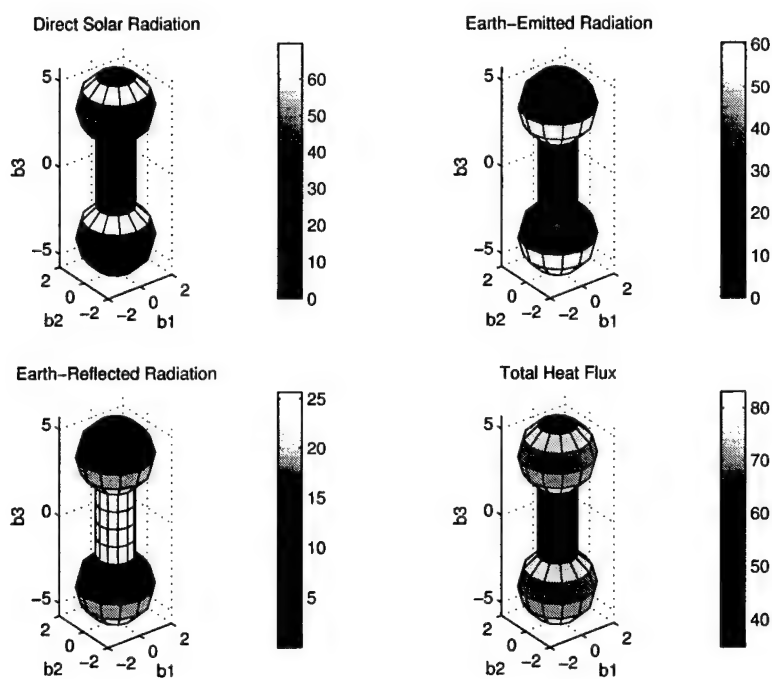


Figure B.2    Heat Flux (W), h=500km, $\eta = 0°$
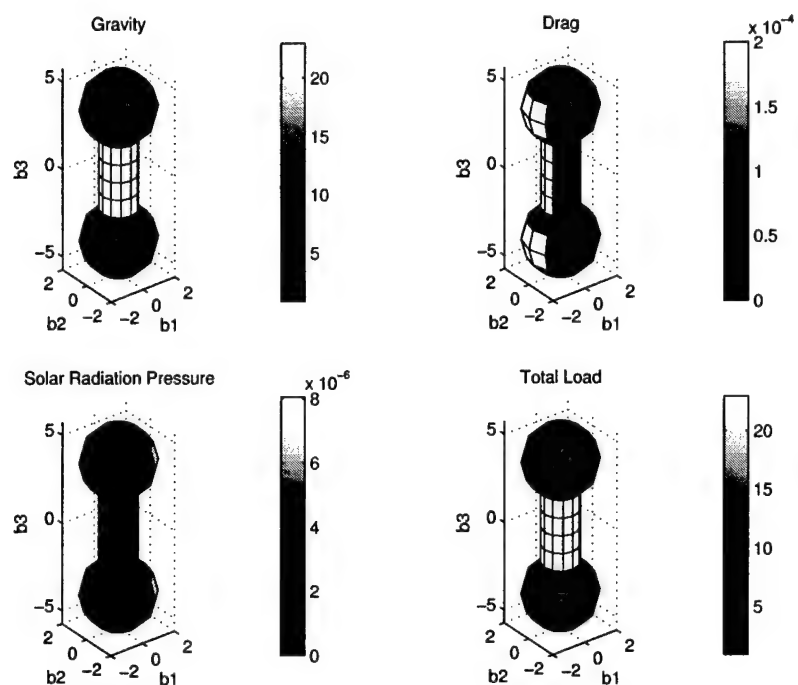
Figure B.3    Mechanical Loads (Pa), h=500km, $\eta = 90°$
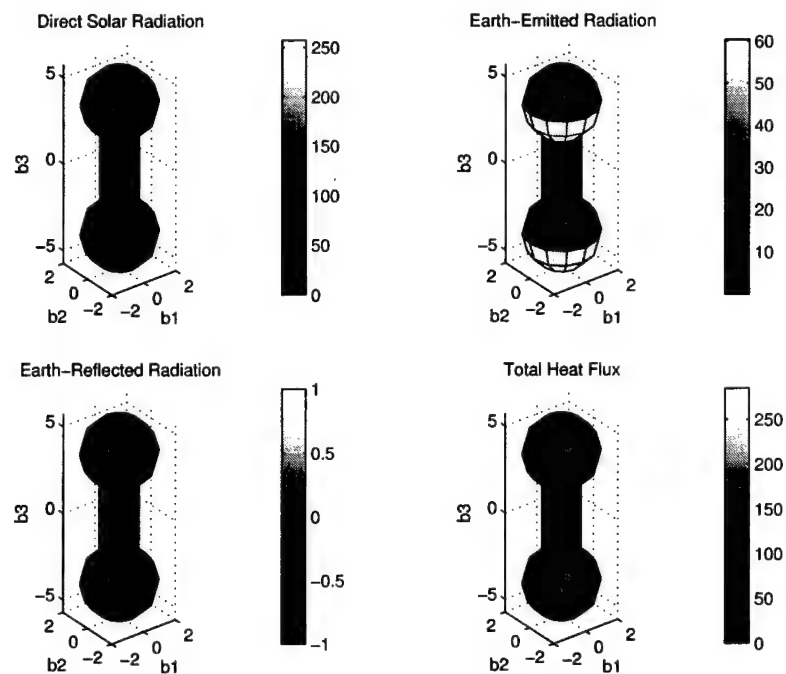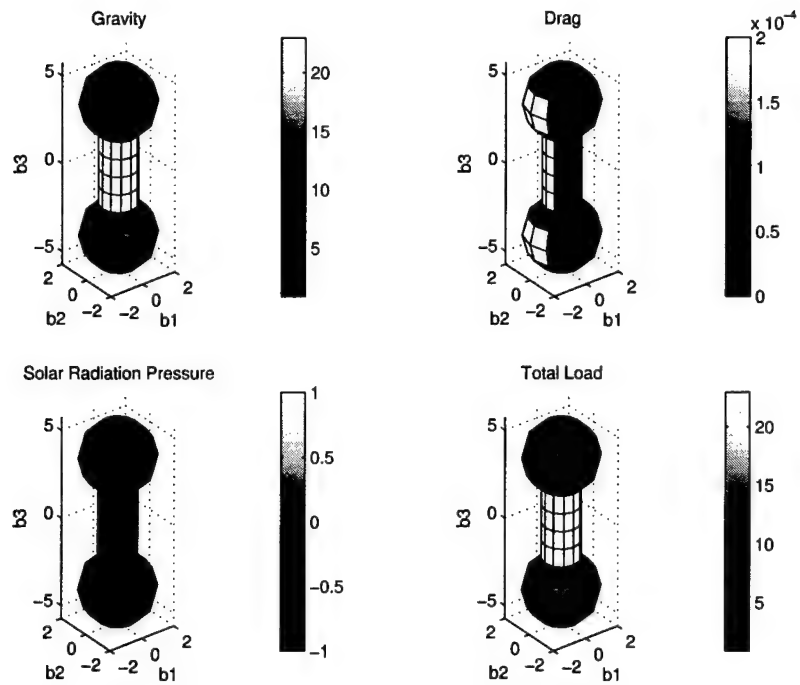


Figure B.4    Heat Flux (W), h=500km, $\eta = 90°$
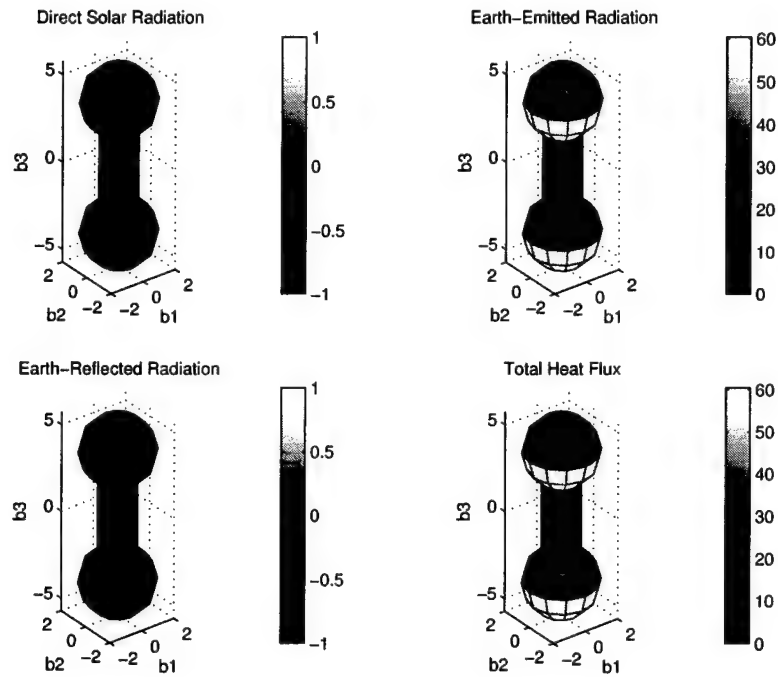
Figure B.5    Mechanical Loads (Pa), h=500km, $\eta = 180°$



Figure B.6    Heat Flux (W), h=500km, $\eta = 180°$

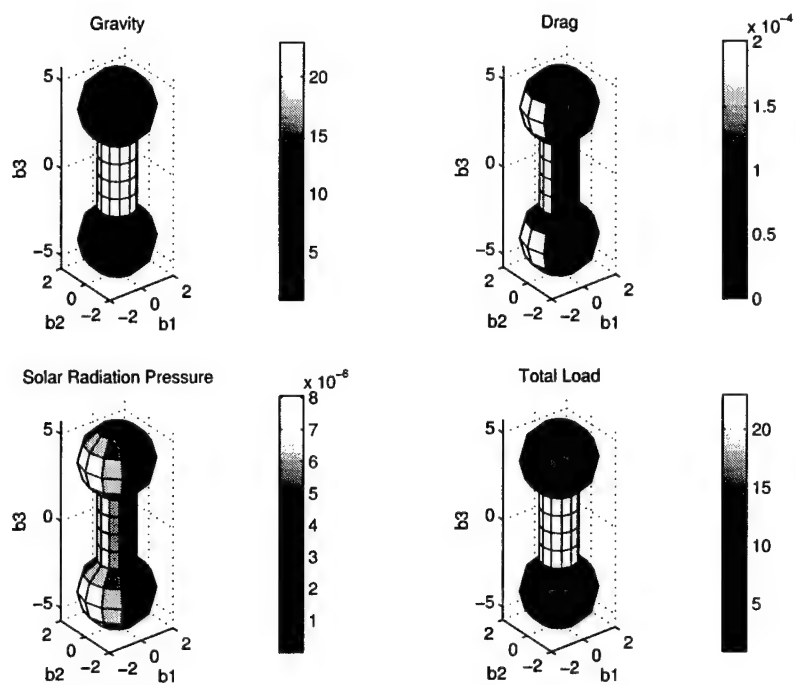Figure B.7    Mechanical Loads (Pa), h=500km, $\eta = 270°$
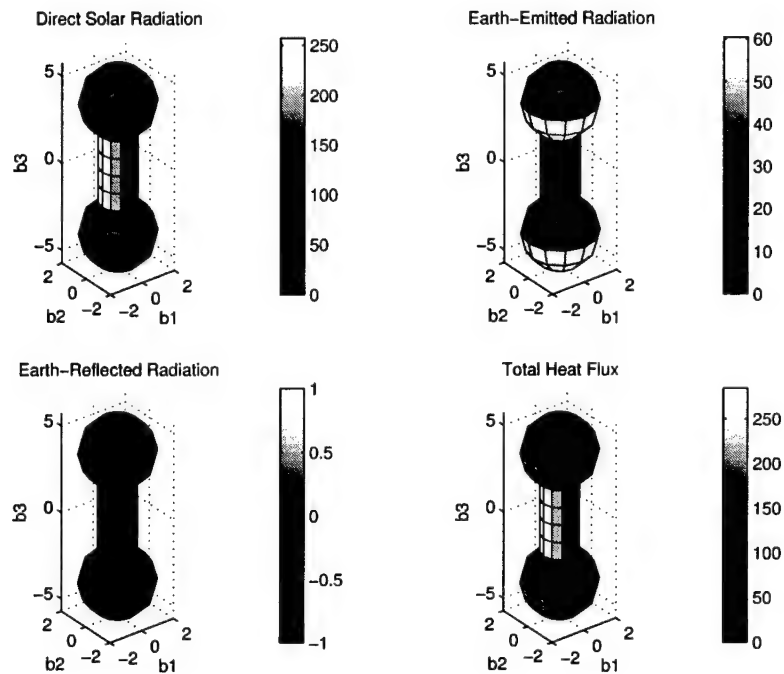


Figure B.8    Heat Flux (W), h=500km, $\eta = 270°$
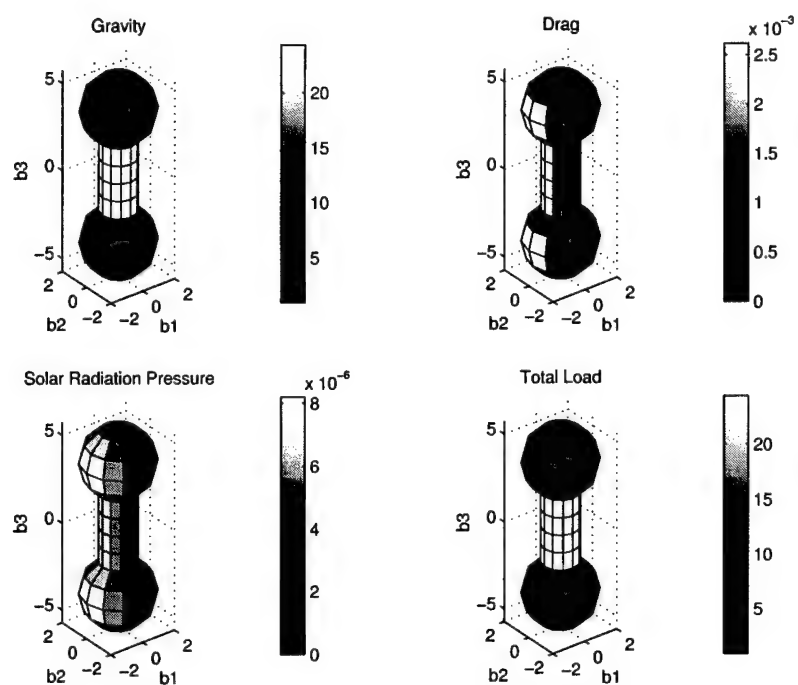
Figure B.9    Mechanical Loads (Pa), h=300km, $\eta = 270°$
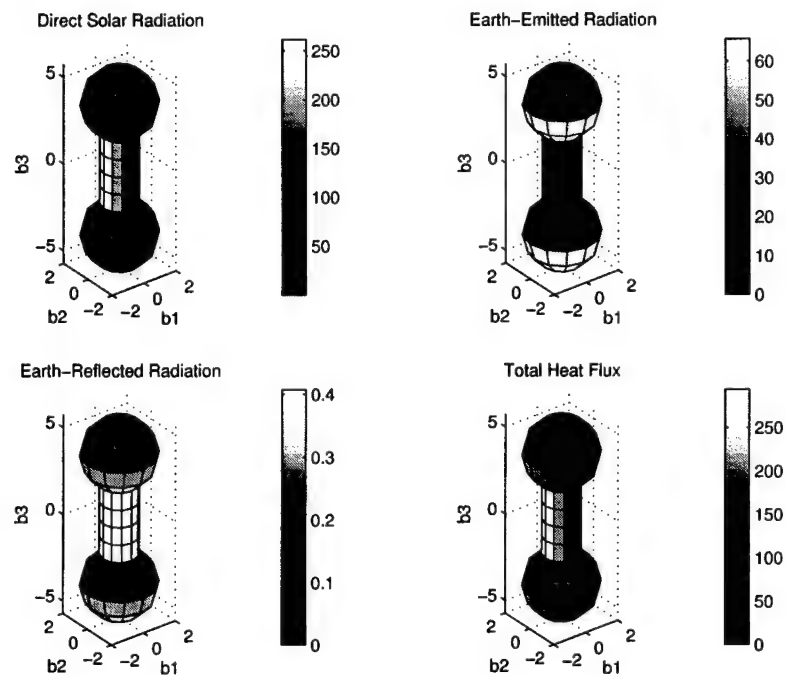


Figure B.10    Heat Flux (W), h=300km, $\eta = 270°$
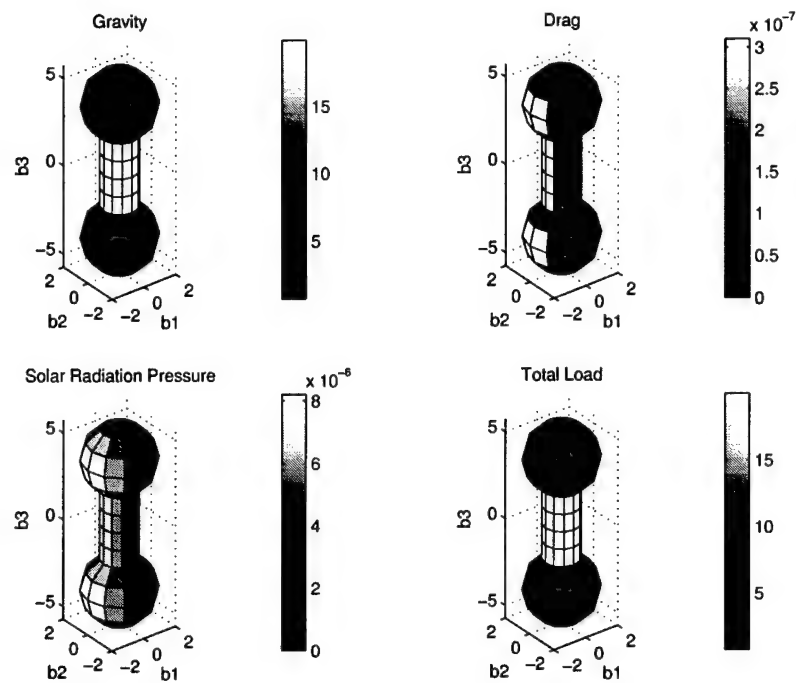
Figure B.11    Mechanical Loads (Pa), h=1000km, $\eta = 270°$
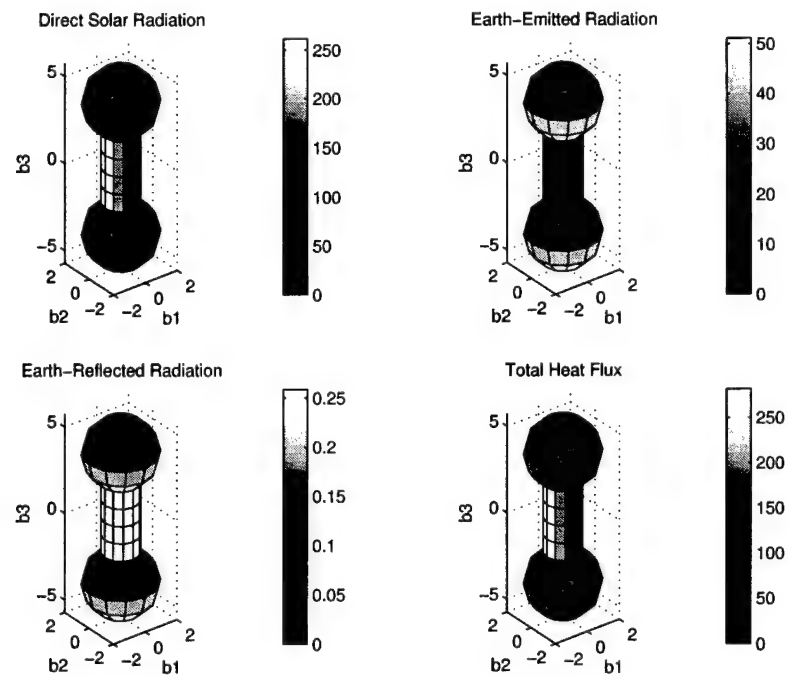


Figure B.12    Heat Flux (W), h=1000km, $\eta = 270°$
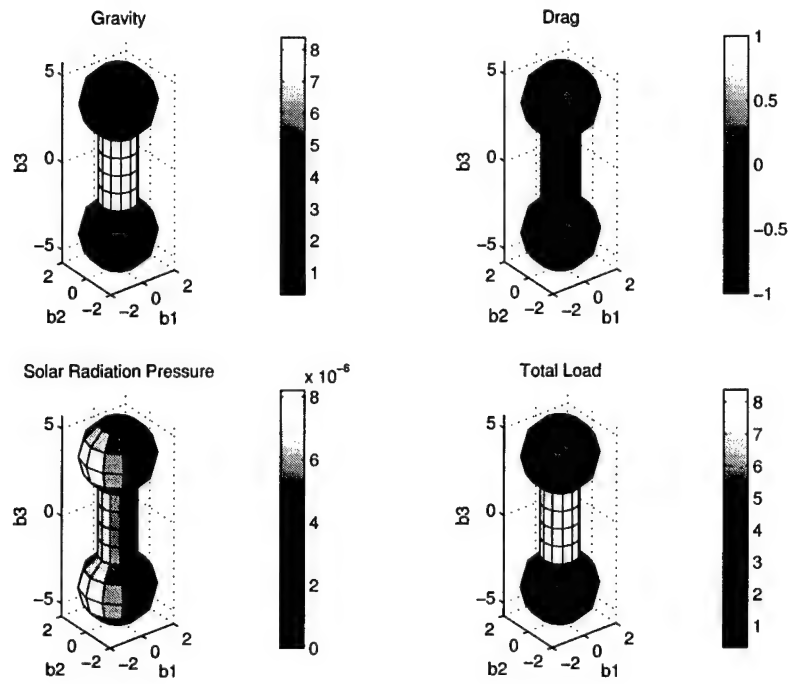
Figure B.13    Mechanical Loads (Pa), h=5000km, $\eta = 270°$
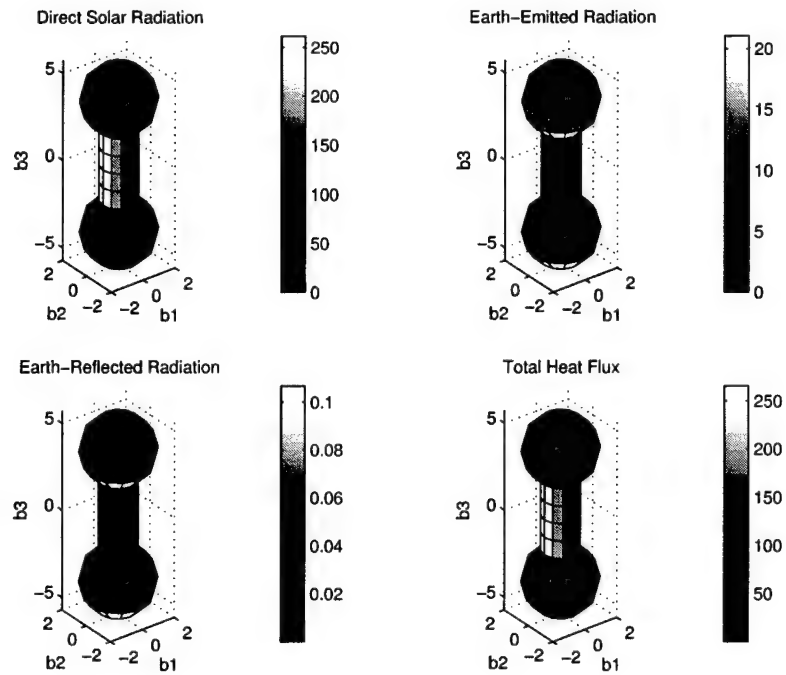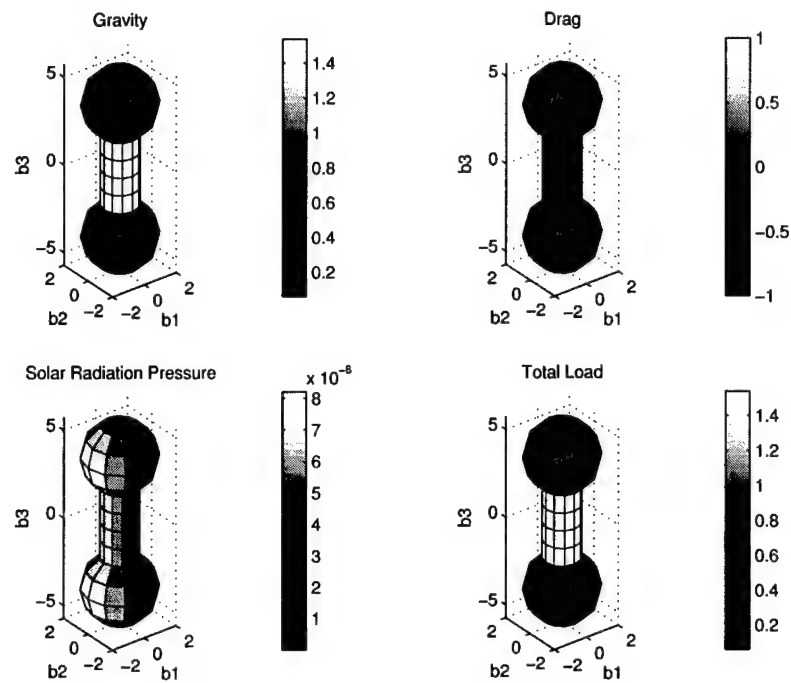


Figure B.14    Heat Flux (W), h=5000km, $\eta = 270°$

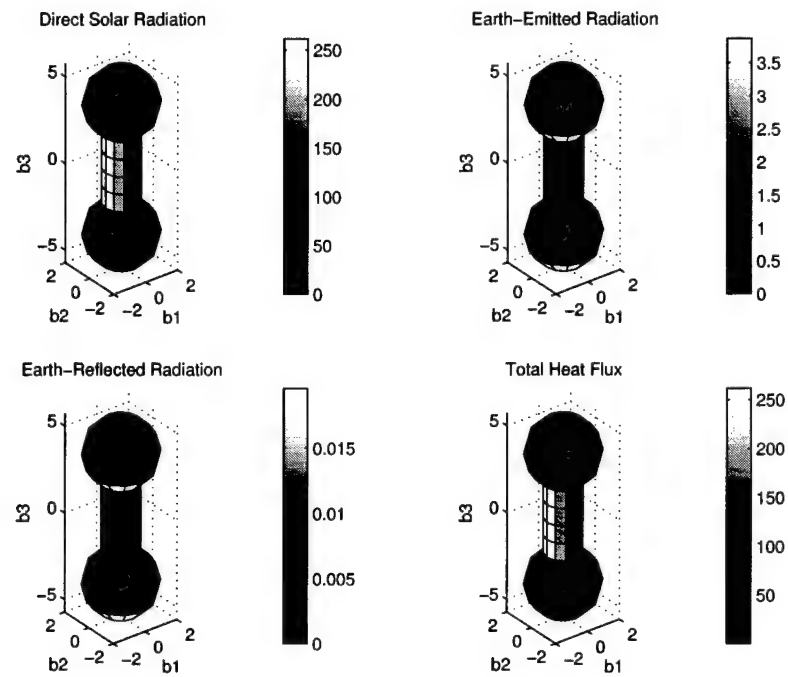Figure B.15     Mechanical Loads (Pa), h=20,183km, $\eta = 270°$



Figure B.16     Heat Flux (W), h=20,183km, $\eta = 270°$

# Appendix C. IAE Mass Model Properties

In order to calculate the torques on the IAE model presented in Chapter 2, the body center of mass and inertia matrix must be determined. This requires knowledge of the center of mass and inertia matrix for each individual component of the structure. In this case, the body is broken into six basic components: the Spartan bus, the mylar lenticular structure, a rigid torus, and three cylindrical struts. A body fixed reference frame is initially set up with its orgin at the Spartan center of mass. The model is illustrated below in Figure C.1.
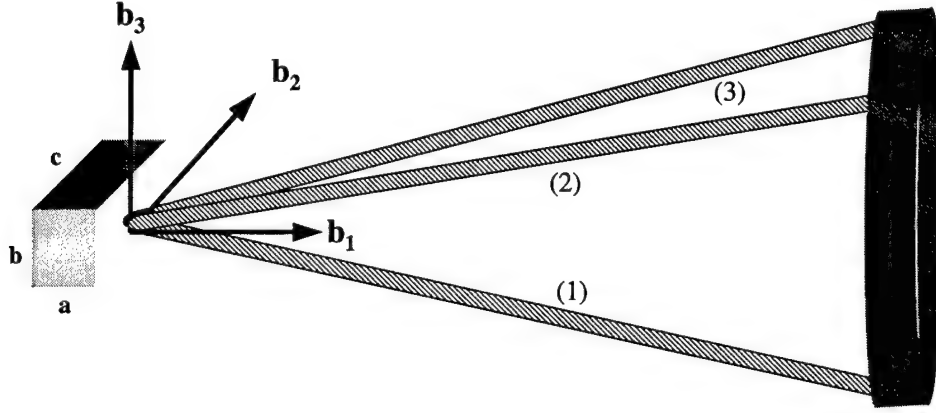


Figure C.1    IAE Model

The Spartan bus is modeled as a 900kg rectangular box with dimensions a=1m, b=3.2m and c=1.2m. Given these measurements, the Spartan inertia matrix about its center of mass is calculated by

$$I^c_{spartan} = \frac{m}{12}\begin{bmatrix} b^2 + c^2 & 0 & 0 \\ 0 & a^2 + c^2 & 0 \\ 0 & 0 & a^2 + b^2 \end{bmatrix} = \begin{bmatrix} 876 & 0 & 0 \\ 0 & 183 & 0 \\ 0 & 0 & 843 \end{bmatrix} kgm^2 \qquad (C.1)$$

The Spartan center of mass is located at the orgin of the defined reference frame.

Each of the struts is modeled by cylindrical shell. The mass of each strut is approximated as follows

$$m_{strut} = \pi * (r_o^2 - r_i^2) * l * \rho \qquad (C.2)$$

where

$$
\begin{aligned}
r_o &= \text{outer radius} = 0.18m \\
r_i &= \text{inner radius} = 0.1797m \\
l &= \text{length} = 28m \\
\rho &= \text{density} = 1360kg/m^3
\end{aligned}
$$

This gives a mass for each strut of 12.9kg. The inertia matrix for each strut can now be calculated.

$$
I_{strut}^c =
\begin{bmatrix}
mr^2 & 0 & 0 \\
0 & \frac{1}{2}mr^2 + \frac{1}{12}ml^2 & 0 \\
0 & 0 & \frac{1}{2}mr^2 + \frac{1}{12}ml^2
\end{bmatrix}
=
\begin{bmatrix}
0.418 & 0 & 0 \\
0 & 843 & 0 \\
0 & 0 & 843
\end{bmatrix}
kgm^2
$$

$$(C.3)$$

This equation is based on the assumption that the strut length runs along the $\hat{b}_1$ axis. In reality, each strut is canted at an angle off of this axis. The symmetry of strut placement though, is such that any off-axis mass terms will cancel when the inertia matrices are added. The moments of inertia will alter slighly due to this small angle, but not enough to significantly degrade the accuracy of the final result. However, this angle must be taken into account when locating the center of mass for each strut. The geometry is displayed in Figure C.2. Each strut is 28m long. For simplicity, assume each strut begins at the Spartan center of mass and connects to the torus at the far end. Consequently, the far end of each strut is displaced approximately 7.3m from the $\hat{b}_1$ axis, as shown in Figure C.2(a). Using the pythagorean theorem,
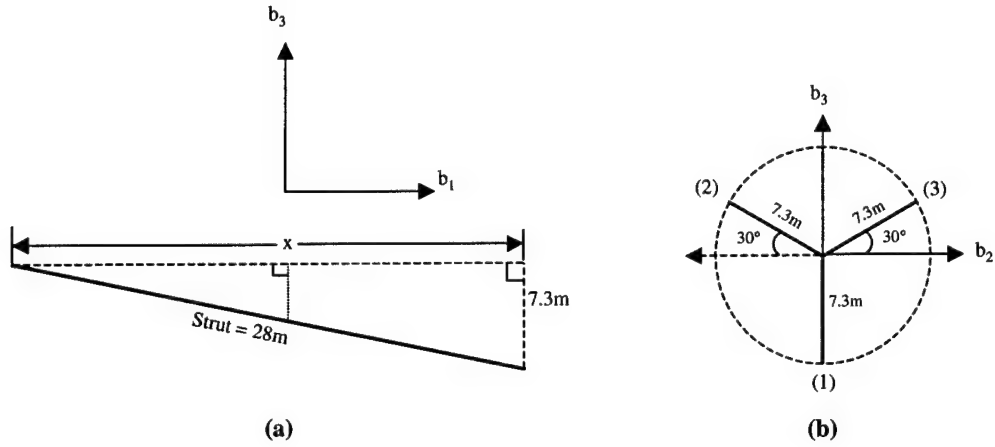
Figure C.2    Strut CM Calculations

the length of each strut along the $\hat{b}_1$, denoted by $x$, is equal to 27.0m. The projected length in the $\hat{b}_2$ and $\hat{b}_3$ directions, denoted by by $y$ and $z$ respectively, is different for each strut. Assume that the struts are spaced at 120 deg intervals with strut (1) lying in the $b_1 - -b_3$ plane, as illustrated in Figure C.2(b). The $y$ and $z$ coordinates of the far ends are given by

$$
\begin{aligned}
y(1) &= 0 \\
z(1) &= -7.3m \\
y(2) &= -7.3m * \cos 30° = -6.32m \\
y(3) &= 7.3m * \cos 30° = 6.32m \\
z(2) &= z(3) = 7.3m * \sin 30° = 3.65m
\end{aligned}
\tag{C.4}
$$

Since the centroid for each strut lies halfway down its length, the center of mass is found by dividing the far end coordinates by two.

$$
cm(1) = \left\{ \begin{array}{c} 13.5 \\ 0 \\ -3.65 \end{array} \right\} m \quad
cm(2) = \left\{ \begin{array}{c} 13.5 \\ -3.16 \\ 1.825 \end{array} \right\} m \quad
cm(3) = \left\{ \begin{array}{c} 13.5 \\ 3.16 \\ 1.825 \end{array} \right\} m \tag{C.5}
$$

The two mylar canopies which make up the lenticular structure are modeled as a single laminar disk with the combined thickness of the two canopies. The mass of the disk is calculated by

$$m_{disk} = \pi * r^2 * t * \rho \tag{C.6}$$

where

$$r = \text{disk radius} = 7m$$

$$t = \text{disk thickness} = 12.7\mu$$

$$\rho = \text{density} = 1400kg/m^3$$

The total mass the lenticular structure is 2.7kg. The center of mass located at $27.3\,\hat{b}_1$ and the inertia matrix is given by

$$I_{disk}^c = \frac{m}{4}\begin{bmatrix} 2r^2 & 0 & 0 \\ 0 & r^2 & 0 \\ 0 & 0 & r^2 \end{bmatrix} = \begin{bmatrix} 66.2 & 0 & 0 \\ 0 & 33.1 & 0 \\ 0 & 0 & 33.1 \end{bmatrix} kgm^2 \tag{C.7}$$

Finally, the rigid torus which supports the mylar canopies is represented, of course, by a torus. This torus has a radius of curvature of approximately 7.3 meters and a cross-sectional diameter of 0.61 meters. Given that the total mass of the inflatable structure is 60kg, subtracting out the mass of the struts and canopies sets the mass of the torus at 18.6kg. The torus center of mass is co-located with the laminar disk center of mass at $27.3\,\hat{b}_1$. The torus inertia matrix is given by

$$I_{torus}^c = \begin{bmatrix} 2mr^2 + \frac{3}{2}ma^2 & 0 & 0 \\ 0 & mr^2 + \frac{5}{4}ma^2 & 0 \\ 0 & 0 & mr^2 + \frac{5}{4}ma^2 \end{bmatrix} = \begin{bmatrix} 1985 & 0 & 0 \\ 0 & 993 & 0 \\ 0 & 0 & 993 \end{bmatrix} kgm^2$$
$$\tag{C.8}$$

where $r$ is the radius of curvature and $a$ is the cross-sectional radius.

Now that the mass properties of each component are known, the center of mass and moments of interia for the entire assembly can be calculated. The center of mass is determined by treating each component as a particle of equal mass, located at the component centroid. The center of mass for a system of particles is calculated by

$$_o r^{cm} = \frac{1}{M} \sum_{i=1}^{n} m_i r_i \qquad \text{(C.9)}$$

where

$$_o r^{cm} \ = \ \text{center of mass position relative to orgin of current reference frame}$$
$$M \ = \ \text{total system mass}$$
$$n \ = \ \text{number of particles}$$
$$m_i \ = \ \text{mass of the ith particle}$$
$$r_i \ = \ \text{location of ith particle}$$

For this model, n is equal to six and the total system mass is 960kg. Taking the product of each component's mass and centroid location and summing over the six components yields the composite body center of mass.

$$r^{cm} = \left\{ \begin{array}{c} 1.15 \\ 0 \\ 0 \end{array} \right\} m \qquad \text{(C.10)}$$

The body center of mass is shifted 1.15 meters along the $\hat{b}_1$ axis from Spartan center of mass.

Summing the inertia matrices of the six components will yield the inertia matrix for the composite body. However, in order to sum inertia matrices, they must all be expressed in a common reference frame and about a common point. All of the inertia matrices are expressed in terms of the body-fixed reference frame defined

above. However, they do not share a common reference point. Each component's inertia is expressed relative to its own center of mass. The inertia matrix for each component about a common reference point can be determined using the parallel axis theorem

$$I^* = I^c + m * \begin{bmatrix} \Delta y^2 + \Delta z^2 & -\Delta y \Delta x & -\Delta z \Delta x \\ -\Delta x \Delta y & \Delta x^2 + \Delta z^2 & -\Delta z \Delta y \\ -\Delta x \Delta z & -\Delta y \Delta z & \Delta x^2 + \Delta y^2 \end{bmatrix} \tag{C.11}$$

where $I^*$ is the inertia about the new reference point, $I^c$ is the inertia about the component's center of mass, and $m$ is the mass of the component. $\Delta x$, $\Delta y$ and $\Delta z$ are the $\hat{b}_1$, $\hat{b}_2$ and $\hat{b}_3$ components of the radius vector from the component center of mass to the new reference point. In this case, the composite body center of mass will be the new reference point. The resulting $\Delta x$, $\Delta y$ and $\Delta z$ values for each component are summarized in Table C.1. Plugging these values into equation (C.11) results in an inertia matrix for each component about the composite body center of mass.

$$I_{spartan} = \begin{bmatrix} 876 & 0 & 0 \\ 0 & 1373 & 0 \\ 0 & 0 & 2033 \end{bmatrix} kgm^2 \qquad I_{canopy} = \begin{bmatrix} 66.2 & 0 & 0 \\ 0 & 1880 & 0 \\ 0 & 0 & 1880 \end{bmatrix} kgm^2$$

$$I_{torus} = \begin{bmatrix} 1985 & 0 & 0 \\ 0 & 13715 & 0 \\ 0 & 0 & 13715 \end{bmatrix} kgm^2 \qquad I_{strut(1)} = \begin{bmatrix} 173 & 0 & 582 \\ 0 & 2982 & 0 \\ 582 & 0 & 2810 \end{bmatrix} kgm^2$$

$$I_{strut(2)} = \begin{bmatrix} 176 & 510 & -291 \\ 510 & 2853 & 75 \\ -291 & 75 & 2943 \end{bmatrix} kgm^2 \qquad I_{strut(3)} = \begin{bmatrix} 176 & -510 & -291 \\ -510 & 2853 & -75 \\ -291 & -75 & 2943 \end{bmatrix} kgm^2$$

All of the component inertia matrices are now expressed about a common reference

Table C.1    Reference Point Displacements

| Component | $\Delta x \, (m)$ | $\Delta y \, (m)$ | $\Delta z \, (m)$ |
|-----------|--------|--------|---------|
| Spartan | 1.15 | 0 | 0 |
| Torus | -26.15 | 0 | 0 |
| Disk | -26.15 | 0 | 0 |
| Strut (1) | -12.35 | 0 | 3.65 |
| Strut (2) | -12.35 | 3.16 | -1.825 |
| Strut (3) | -12.35 | -3.16 | -1.825 |

point. Summing these matrices yields the moments of inertia for the composite body.

$$I_{IAE}^c = \begin{bmatrix} 3452 & 0 & 0 \\ 0 & 25,656 & 0 \\ 0 & 0 & 26,324 \end{bmatrix} kgm^2 \tag{C.12}$$

Note that the defined reference frame constitutes the principle axes. The body is nearly axisymmetric, with the minimum moment of inertia occurring about the $\hat{b}_1$ axis and the maximum moment of inertia along the $\hat{b}_3$ axis.

# Bibliography

1.  Alexander, Margaret B. *The Natural Space Environment: Effects on Spacecraft*. Research Publication NASA-RP-1350, Marshall Space Flight Center AL: National Aeronautics and Space Administration, November 1994.

2.  Bowman, W. Jerry, Kenneth A. Carpenter and Christopher P. Chaplin. "Nonchemical Propulsion." Written by the Aerospace Engineering Department at the Air Force Institute of Technology, Wright-Patterson AFB OH, Spring 1994.

3.  Cassapakis, Costa and Mitch Thomas. "Inflatable Structures Technology Development Overview." *AIAA Space Programs and Technologies Conference, Huntsville, AL, Sept. 26–28, 1995*. New York: American Institute of Aeronautics and Astronautics, 1995.

4.  Chobotov, Vladimir A. *Spacecraft Attitude Dynamics and Control*. Malabar FL: Krieger Publishing Company, 1991.

5.  Chobotov, Vladimir A. *Orbital Mechanics* (Second Edition). AIAA Education Series, Reston VA: American Institute of Aeronautics and Astronautics, 1996.

6.  Criswell, Marvin E., et al. "Design and Performance Criteria for Inflatable Structures in Space." *Engineering Construction and Operations in Space2*. Proceedings of the Fifth International Conference on Space, edited by Stewart W. Johnson. 1045–1051. New York: ASCE, 1996.

7.  Dornheim, Michael A. "Inflatable Structures Taking to Flight," *Aviation Week and Space Technology*, *150*(4):60–62 (Jan 1999).

8.  Friese, Michael, "Spartan/IAE." L'Garde homepage, http://www.lgarde.com /programs/iae.html, October 2000.

9.  Garrett, Henry B. and Charles P. Pike, editors. *Space Systems and Their Interactions with Earth's Space Environment*, *71*. Progress in Astronautics and Aeronautics. New York: American Institute of Aeronautics and Astronautics, 1980.

10. Glassner, Andrew S., editor. *An Introduction to Ray Tracing*. San Diego: Academic Press, 1989.

11. Hedgepeth, John M. *Critical Requirements for the Design of Large Space Structures*. Contract Report NASA-CR-3484, Washington DC: National Aeronautics and Space Administration, November 1981.

12. Howell, J.R. *A Catalog of Radiation Configuration Factors*. New York: McGraw-Hill, 1982.

13. Jacchia, L.G. *New Static Models of the Thermosphere and Exosphere with Empirical Temperature Profiles*. SAO Special Report 313, Cambridge MA: Smithsonian Astrophysical Observatory, 1970.

14. Johnston, John D. and Earl A. Thorton. "Thermally Induced Dynamics of Satellite Solar Panels," *Journal of Spacecraft and Rockets*, *37*(5):604–613 (Sep-Oct 2000).

15. Killeen, T.L., et al. "Modeling and Prediction of Density Changes and Winds Affecting Spacecraft Trajectories." *Environmental Effects on Spacecraft Positioning and Trajectories 73*. Geophysical Monograph Series, edited by A. Vallance Jones. 83–108. Washington DC: American Geophysical Union, 1993.

16. Logan, Daryl L. *A First Course in the Finite Element Method*. Boston: PWS-Kent Publishing, 1986.

17. "MatWeb." The Online Materials Information Resource, Automation Creations Inc., http://www.matweb.com, October 2000.

18. Meeus, Jean. *Astronomical Algorithms* (Second Edition). Richmond VA: Willmann-Bell Inc., 1998.

19. Meriam, J.L. and L.G. Kraige. *Engineering Mechanics* (Fourth Edition), *2.* Dynamics. New York: John Wiley and Sons Inc., 1997.

20. Piscane, Vincent L. and Robert C. Moore, editors. *Fundamentals of Space Systems*. New York: Oxford University Press, 1994.

21. Robie, R.G. "Atmospheric Drag." *Advances in the Astronautical Sciences: Guidance and Control 1991 74*. Proceedings of the Annual Rocky Mountain Guidance and Control Conference, Keystone, CO, Feb. 2–6, 1991, edited by Robert D. Culp and James P. McQuery. 611–626. San Diego: Univelt Inc., 1991.

22. Rosborough, George W. "Gravitational Effects on Low-Earth Orbiters." *Advances in the Astronautical Sciences: Guidance and Control 1991 74*. Proceedings of the Annual Rocky Mountain Guidance and Control Conference, Keystone, CO, Feb. 2–6, 1991, edited by Robert D. Culp and James P. McQuery. 587–610. San Diego: Univelt Inc., 1991.

23. Satter, Celeste M. and Robert E. Freeland. "Inflatable Structures Technology Applications and Requirements." *AIAA Space Programs and Technologies Conference, Huntsville, AL, Sept. 26–28, 1995*. New York: American Institute of Aeronautics and Astronautics, 1995.

24. *Solar Events: Solar Eclipse Predictions*. Technical Order CG-SCF-225C, Schriever AFB CO: Third Space Operations Squadron, December 1989.

25. "Spartan 207 Mission." Excerpt from NASA's Spartan homepage, http://spartans.gsfc.nasa.gov, September 2000.

26. Tascione, Thomas F. *Introduction to the Space Environment* (Second Edition). Malabar FL: Krieger Publishing Company, 1994.

27. Thorton, Earl A. and Richard S. Foster. "Dynamic Response of Rapidly Heated Space Structures." *Computational Nonlinear Mechanics in Aerospace Engineering 146*. Progress in Astronautics and Aeronautics, edited by Satya N. Atluri. 451–477. Washington DC: American Institute of Aeronautics and Astronautics, 1992.

28. Tragesser, Steven G. Class notes, MECH 533, Intermediate Spaceflight Dynamics. School of Engineering and Management, Air Force Institute of Technology, Wright-Patterson AFB OH, Spring Quarter 2000.

29. Wertz, James R. *Spacecraft Attitude Determination and Control*. Boston: D. Reidel Publishing Co., 1978.

30. Wiesel, William E., Jr. *Spaceflight Dynamics* (Second Edition). Boston: Irwin McGraw-Hill, 1997.

31. Zimbelman, Darrell F. "Thermal Elastic Shock and its Effect on TOPEX Spacecraft Attitude Control." *Advances in the Astronautical Sciences: Guidance and Control 1991 74*. Proceedings of the Annual Rocky Mountain Guidance and Control Conference, Keystone, CO, Feb. 2–6, 1991, edited by Robert D. Culp and James P. McQuery. 311–334. San Diego: Univelt Inc., 1991.

## *Vita*

Captain Donald J. Davis graduated from Glenbard North High School in Carol Stream, Illinois in June 1988. That same month, he entered undergraduate studies at the United States Air Force Academy in Colorado Springs, Colorado. On 27 May 1992 he graduated with a Bachelor of Science degree in Engineering Mechanics and received his Regular Commission.

Captain Davis's first assignment was to the First Space Operations Squadron, Falcon AFB, Colorado, where performed command and control operations for the Defense Support Program and Defense Meteorological Satellite Program. In October 1995 he was assigned to the Twelfth Space Warning Squadron at Thule AB, Greenland. There he performed duties as a crew commander and chief of training for missile warning and space surveillance operations. Following his return to CONUS in October 1996, he was assigned to the Tenth Missile Squadron, Malmstrom AFB, Montana, where he served as a Minuteman III Missile Combat Crew Commander and evaluator. In August 1999, he entered the Graduate School of Engineering and Management, Air Force Institute of Technology. Upon graduation, he will be assigned to the Space Warfare Center, Schriever AFB, Colorado.

| REPORT DOCUMENTATION PAGE | | Form Approved OMB No. 0704-0188 |
|---|---|---|

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
**PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

| 1. REPORT DATE (DD-MM-YYYY) 08-03-2001 | 2. REPORT TYPE Master's Thesis | 3. DATES COVERED (From - To) Jun 2000 - Mar 2001 |
|---|---|---|

**4. TITLE AND SUBTITLE**
ENVIRONMENTAL DISTURBANCE MODELING FOR LARGE INFLATABLE SPACE STRUCTURES

**5a. CONTRACT NUMBER**

**5b. GRANT NUMBER**

**5c. PROGRAM ELEMENT NUMBER**

**6. AUTHOR(S)**
Davis, Donald J., Captain, USAF

**5d. PROJECT NUMBER**

**5e. TASK NUMBER**

**5f. WORK UNIT NUMBER**

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 P Street, Building 640
WPAFB OH 45433-7765

**8. PERFORMING ORGANIZATION REPORT NUMBER**
AFIT/GSO/ENY/01M-02

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
AFOSR/NA
Attn: Dr. Daniel Segalman
801 N. Randolph St.
Arlington, VA 22203          COMM: (703) 696-7259

**10. SPONSOR/MONITOR'S ACRONYM(S)**

**11. SPONSOR/MONITOR'S REPORT NUMBER(S)**

**12. DISTRIBUTION/AVAILABILITY STATEMENT**

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

**13. SUPPLEMENTARY NOTES**

**14. ABSTRACT**
Tightening space budgets and stagnating spacelift capabilities are driving the Air Force and other space agencies to focus on inflatable technology as a reliable, inexpensive means of deploying large structures in orbit. Recent improvements in rigidization techniques make the use these inflatable structures feasible for a growing number of missions. For many of these missions, the primary design requirement is dimensional accuracy of the structure. Finite element analysis offers a means of predicting structural behavior in orbit. The analysis requires knowledge of external loads. This thesis examines the environmental disturbances which act upon large, orbiting structures. Calculations are made on a base model to relate the torques generated by these disturbances to the orbital altitude. This facilitates identification of the critical loads. An environmental disturbance model is then developed in MATLAB. The model calculates the critical loads on each element of a faceted structure as it propagates through its orbit. A basic structure is defined and run through the model. Results and analysis for various orbits are presented to verify accuracy of the code and validate the derived torque-altitude relationships.

**15. SUBJECT TERMS**
Inflatable Structures, Critical Loads, Space Environment, Gravity Gradient, Drag, Thermal Snap, Solar Radiation Pressure, Finite Element Analysis

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON Major Greg Agnes, ENY |
|---|---|---|---|---|---|
| a. REPORT U | b. ABSTRACT U | c. THIS PAGE U | UU | 219 | 19b. TELEPHONE NUMBER (Include area code) (937) 255-6565, ext 4317 |

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18